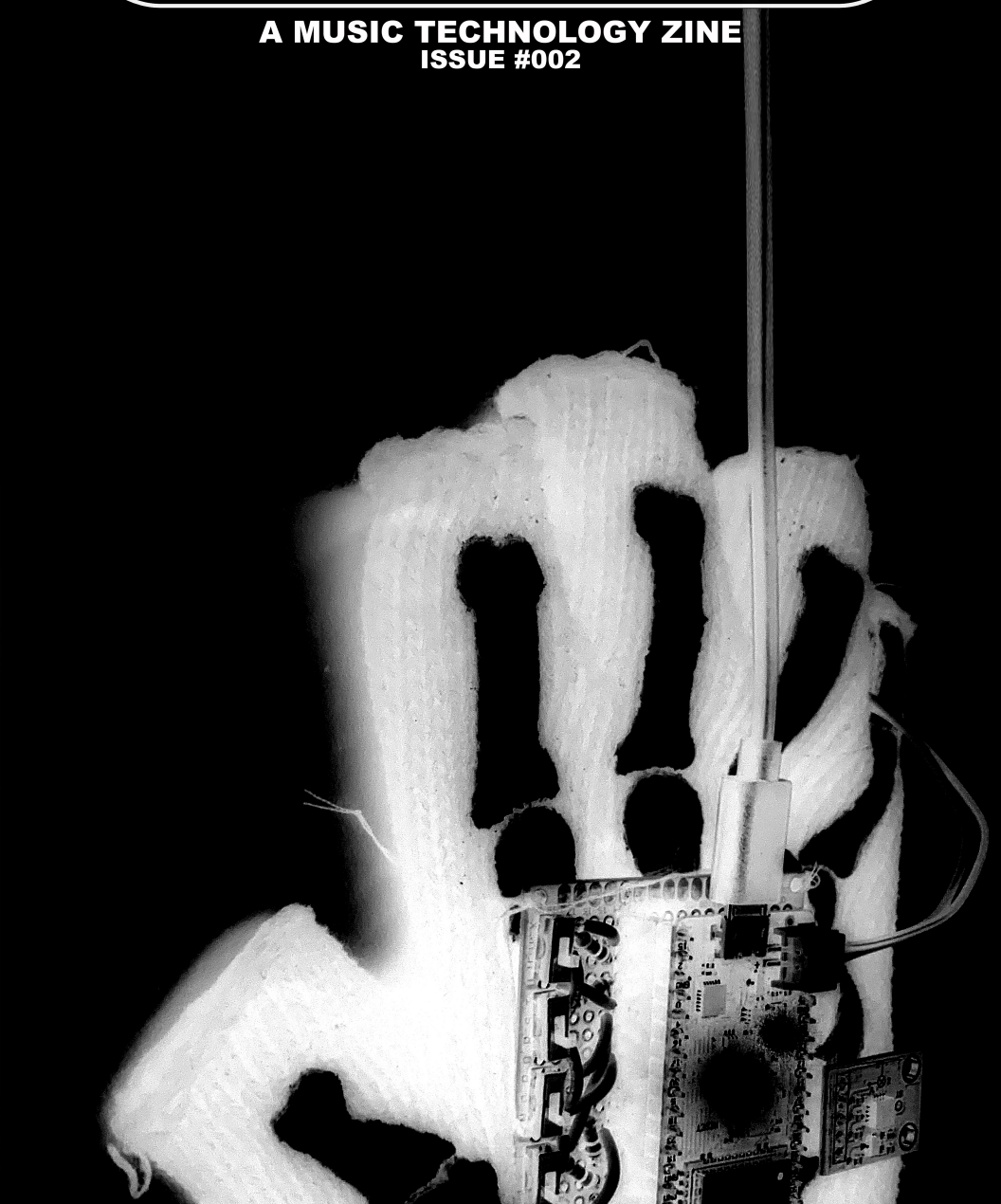
circuit punk

A MUSIC TECHNOLOGY ZINE

ISSUE #002



CONTENTS

Mind Over MIDI (DIY Brain Scanner) 1

Analog XOR 4

Jumperless Adventures 6

Turning Old Radios Into Guitar Amps 9

Les Belles Noiseuses 11

VCO Freak Machines 14

All-Pass Filters 16

Bodybend 18

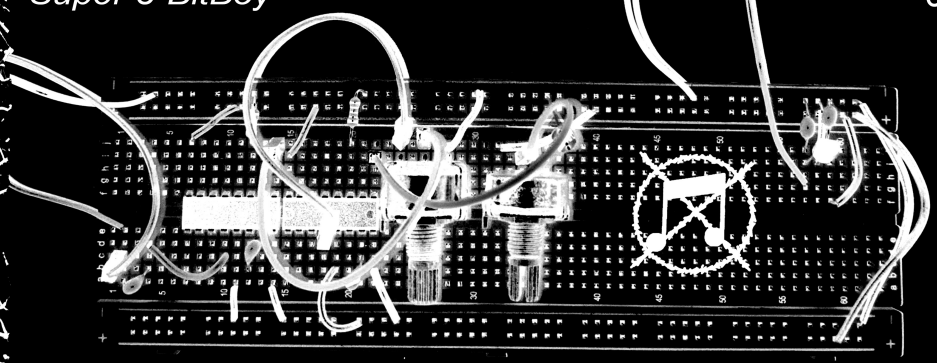
Intervivew: Moritz Klein 21

Simulation Assisted Distillation Of A Circuit's Essence 24

Interview: The Audio Programmer 27

Shoving A Pedal Into A Eurorack 30

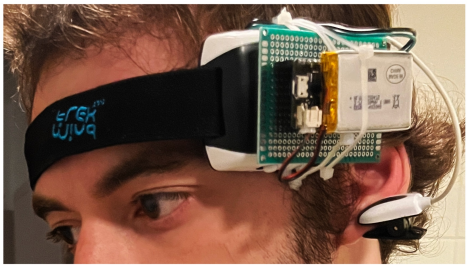
Super 8-BitBoy 36



MIND OVER MIDI
DIY BRAINWAVE SENSOR

Cameron Glass
cameron@circuitpunk.org

When I was little, I would constantly see commercials on TV for a game called Mindflex, where a ball is “levitated” through an obstacle course by a fan nozzle. The air pressure of the nozzle is controlled by a brainwave-sensing headband worn by the player. I always thought it looked amazing, but had some skepticism about the whole “telekinesis” aspect. Is that kind of technology really…in a toy? I wondered. Now as an adult, I basically wonder the same thing. But now, I’m ready to put it to the test (in a musical context!) There’s many used Mindflex headsets on eBay for around $20. That’s way cheaper than any EEG headset on the market. It’s even cheaper than the brainwave sensor inside of the headset. And it’s a perfect candidate for a super affordable, Bluetooth, brainwave-sensing MIDI controller.



Mindflex MIDI controller

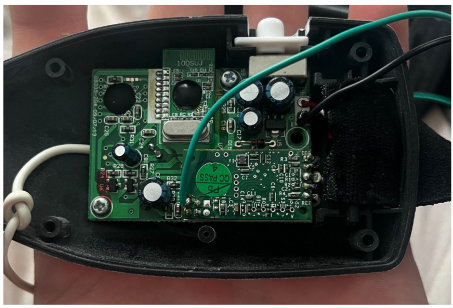
Expectations

Mindflex uses a chip by Neurosky that has been used in two mass-market products: Mindflex and the Star Wars Force Trainer. I’m not a neuroscientist, but even I can tell that having a single forehead electrode on this rinky dink 15+ year old toy is no match for the multi-electrode EEG caps used in the medical field. There’s also something called “wet” electrodes which provide less resistance between the electrode and your skin, and which Mindflex does not use. The dry electrode setup should still give us some weak brain signals to work with, albeit with a lot of noise baked in.

Hardware/Software Basics

Thankfully I’m not the first to try to circuit bend these toys. As usual, I’m standing on the shoulders of giants: I found a tutorial from 2010 by Frontiernerds[1] that walks through the hardware and software aspects of getting brain data onto an Arduino compatible microcontroller. In this case I’m using an ESP32 WROOM 32 dev board, since I’m keeping Bluetooth capability in mind. Their article is rich in detail, and I highly recommend reading it for anyone interested in this type of project. But I’ll summarize the relevant portions.

Pop open the side panels (one is for batteries) and locate the larger of the two daughter boards. One wire gets soldered to the transmit pin (labeled ‘T’, next to ‘R’), and another wire gets soldered to ground (black wire on the main board). The transmit pin gets connected to an Rx pin on the Arduino, and the ground to ground. Cut or drill a little divot in the case in order to put it neatly back together. I’m clearly not a fabrication guy- I slapped this together with cable ties and dreams.



As for the software, an Arduino library by Erik Mika[2] parses the raw byte data from the headset and returns it in a convenient format. A few example sketches are included to get started printing data in the serial monitor.

[1] https://frontiernerds.com/brain-hack
[2] https://github.com/kitschpatrol/Brain

You can now access each of the 8 power-band values of the headset. Each band is associated with a mental state like focus, relaxation, or multi-sensory input. The headset returns data at a rate of about one packet per second.

Musical Mod

To make this more useful in a musical context, let's turn it into a Bluetooth MIDI controller. With the ESP32 and the ESP32-BLE-MIDI library by Maxim André[3], it's quite easy to get working. The idea is to have one of the power-band values smoothed out (interpolated), and sent as a stream of control change messages:

```
1  #include <BLEMIDI_Transport.h>
2  #include <hardware/BLEMIDI_ESP32.h>
3  #include <Brain.h>
4
5  Brain brain(Serial);
6  BLEMIDI_CREATE_INSTANCE("MINDFLEX",MIDI);
7
8  unsigned long lastT = 0;
9  unsigned long currentT = 0;
10 int interval = 0;
11 int lastV = 0;
12 int currentV = 0;
13 float interp = 0.0;
14 int interpV = 0;
15 int mappedV = 0;
16 boolean isFirst = true;
17
18 void setup() {
19   MIDI.begin(10);
20   Serial.begin(9600);
21 }
22
23 void loop() {
24   currentT = millis();
25   if (brain.update()) {
26     if (isFirst) {
27       interval = 1000;
28       isFirst = false;
29     }
30     else {
31       interval = currentT - lastT;
32     }
33     lastT = currentT;
34     lastV = currentV;
35     currentV = brain.readTheta();
36     Serial.println(currentV);
37   }
38   interp = min(int(currentT-lastT),interval)/float(interval);
39   interpV = lastV + int((currentV-lastV)*interp);
40   mappedV = constrain(int(127*interpV/1000000.0),0,127);
41   MIDI.sendControlChange(1, mappedV, 1);
42   delay(50);
43 }
```

The Bluetooth MIDI portion of the project is only a few lines of code (6, 19 and 41). The only function used from this library is *sendControlChange()*, which takes a control number, value, and channel number as arguments.

The brain portion of the project is also quite minimal: It waits for a new packet of data (by checking *brain.update()* repeatedly). Once the function returns *true*, we call *getTheta()* to receive the value of the Theta band, associated with attention/focus.

Finally we interpolate between the values received from the headset:

- Record the interval of milliseconds passed between the last and current capture (lines 26-32)
- Keep track of the previous and current values (lines 34-35)
- Get an interpolated value based on the ratio between milliseconds passed since last iteration and the recorded interval, as well as the difference between the current and previous values. (lines 38-39)
- Map the interpolated value to a MIDI range (line 40)

Troubleshooting Quirks

- Arduino Sketch would only upload to the ESP32 when the Rx pin was disconnected.
- Example sketches from the Brain library would only work when uploaded via Teensyduino rather than vanilla Arduino. For a library that hasn't been maintained in over a decade, this sort of makes sense. I suspect that a very old version of Arduino would also work.

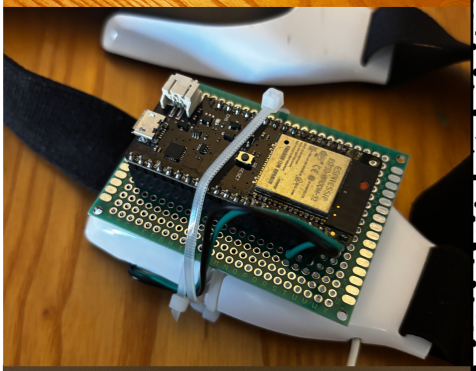
Upgrades

Here's some improvements that I'll leave as exercises to the reader:

- Send data from multiple bands on with different MIDI control numbers

[3] https://github.com/max22-/ESP32-BLE-MIDI

    - Include buttons on the headset to make mapping the different bands easier
- Include a trim pot on the headset to adjust the range of data being mapped
- Average the last X samples in an attempt to get more accurate yet slower data
- Use the *readSignalQuality()* function from the Brain library to send data only when the headset is being worn.
    - Add an LED indicator to the headset for when it is turned on and sending data.
- Use either PWM (pulse-width modulation) or a DAC (digital-audio converter) as well as an op amp boost stage to turn the headset into a Control Voltage generator for controlling synthesizers and guitar pedals,



Cj wearing the Mindflex MIDI controller

Conclusions

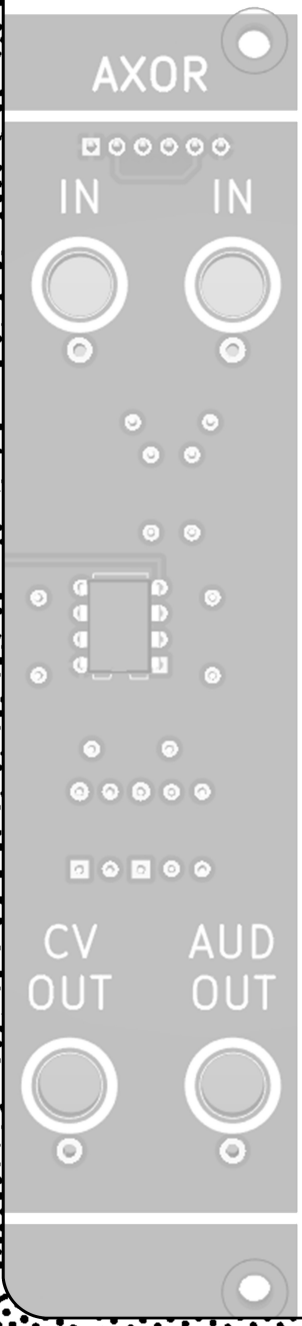
I'm not gonna sugar coat it: It's difficult to tell if this is really working to some small degree or if it's all placebo. It's also annoying to have the packets come in only once every second. But…it is really sending data from a dry electrode on my forehead and Neurosky's TGAM sensor module. And that in itself is pretty cool.

In the future I'd like to try some more sophisticated headsets; Some of them come with fancy features, like the ability to train a machine learning model to recognize certain thoughts or sensory experiences and trigger a custom function in response. Others are simpler, but will give you raw and much more accurate brainwave data.

There you have it. A brain MIDI connection at a low low price. If you wish to try and make this type of project completely from scratch, you may be interested in the Illumino Brainwave hat.

AXOR

by Fihdi



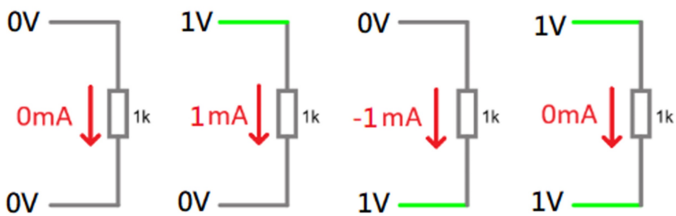
AXOR (short for analog XOR) is an all-analog ringmodulator and a weird CV mixer consisting of only 14 components. Putting two square waves through an XOR gate results in a ringmodulated type of effect, which is something I wanted to hear for myself. I could have just used a standard XOR IC like the CD4070 to do this, but I wanted to create something more exciting. The resulting circuit uses two opamp circuits to recreate the XOR output instead of transistors. Unlike with a traditional logic gate, that can only go high or low, this circuit can create a continuous output signal, so it can be used as a weird audio mixer or a CV source for synthesizers.

How do we make a logic gate out of opamps?

Let us forget about switches, transistors and the formula for the XOR gate in boolean algebra for a second, and just take a look at the truth table:

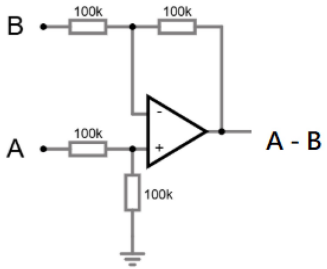
A	B	OUT
0	0	0
0	1	1
1	0	1
1	1	0

Notice how the output is 0 when the inputs are the same and 1 when the inputs differ. This is the exact same behaviour that we get when applying two voltages to the leads of a resistor, and viewing the resulting current as the “output”. If we apply every combination of 1V and 0V to the leads of a 1k resistor we get the following four possible combinations:

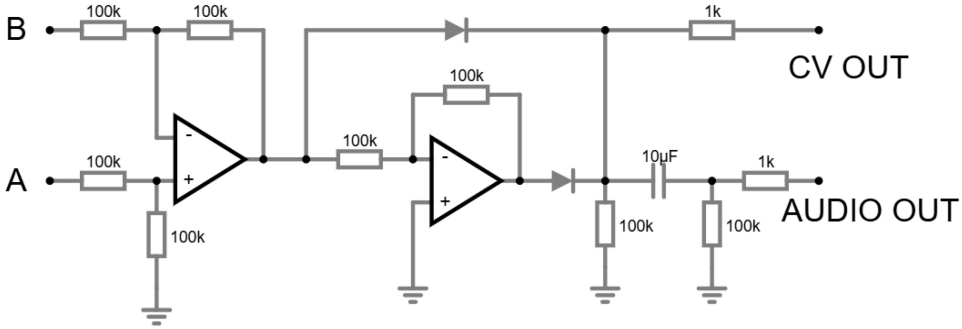


A	B	Current
$0V$	$0V$	0mA
$0V$	1V	1mA
1V	$0V$	-1mA
1V	1V	0mA

The truth table from above already looks like that of an XOR gate. But we want an output voltage, not a current, so we use a subtractor opamp circuit that converts the difference into a voltage.



Now with our 0V and 1V example, our circuit outputs 1V, 0V and -1V, but the truth table of an XOR gate does not have negative values. We gotta stay true to the truth table and flip the -1V to 1V, since it is the absolute magnitude of this current that we view as the output. This can be done with a rectifier circuit that we add to the output of our subtractor. And our circuit is finished!



Since this circuit only outputs positive voltages, we still need to re-center the signal at the audio output to be around 0V, so an AC-highpass is added, as well as a 1k output resistors per convention. With only 14 components we built a ringmodulator that can be used as a weird CV mixer or simply as an XOR gate. The circuit can also be used as a full wave rectifier by only connecting one input to it.

simulator:

Jumperless Adventures

George "Archit3ch" Gkountouras

I've been working on grey-box circuit modeling for audio effects for years. No matter which method you use, whether classical SPICE-based techniques or those newfangled neural networks, one thing remains constant. You can never have enough data!

Introduction

Enter Kevin "Architeuthis Flux" Cappuccio and his crowdfunded breadboard with software-defined connections, the Jumperless v5[1]. The Reddit announcement[2] was split. Many decried it as "too expensive" for the privilege of getting rid of cables. Others could see the vision, their heads instantly filling with possibilities ("wait, it can also do *that* for me!") for dumping chip ROMs, curve tracing, and, sure enough, audio processing. Since you are reading this article, it is likely that you belong in the latter category. ;)

A plan forms

When I read the announcement[3], I thought of my data collection needs, and how this device promised to solve them[4]. Of course, automated data collection is not new. Famously, Neural DSP has a robot[5] that turns knobs on music equipment to standardize tedious measurements. However, this approach presents multiple downsides. For one, it features mechanical parts that invite wear-and-tear. Not ideal when measuring expensive vintage equipment, especially if obtained on loan! Next, the device can only measure inputs and outputs of the system, missing the critical inter-stage bias points. In contrast, the Jumperless can measure at multiple points (including the middle of the circuit during operation!), and can also introduce multiple input signals to aid in your tests. For instance, it can precisely control the power of a specific stage, simulating a dying battery. However, the real power of the Jumperless is that it can *reconfigure* the Device-Under-Test (DUT), adding/removing stages, changing impedances dynamically, flipping bias points, creating feedback paths and so on. I'd like to see a robot do that! Finally, the Terminator approach is patented by Neural DSP, whereas analog switches are not!

Challenges

The plan: script the Jumperless to perform measurements of e.g. guitar pedals, leave it in a closet for a week, and obtain fresh nourishment for our data-hungry models. However, there are some limitations. These hail mostly from the analog crossbars. My first idea was to measure high-voltage vacuum tubes. Lack of self-preservation instinct aside, the box for Jumperless contains a paper with a warning not to put more than +-8V anywhere on the board. Bummer.[6] Another issue is that each analog

---

[1] https://jumperless.org

[2] https://www.reddit.com/r/arduino/comments/1fmxqz6/jumperless_v5_lets_you_prototype_like_a_nerdy/

[3] https://www.crowdsupply.com/architeuthis-flux/jumperless-v5

[4] There are similar projects: https://github.com/jatinchowdhury18/differentiable-wdfs, https://blog.native-instruments.com/the-making-of-icm/

[5] https://neuraldsp.com/quad-cortex-updates/introducing-tina

[6] According to Kevin, a potential future Jumperless v6 will be able to handle high voltages.

switch has 85 ohms of resistance. This can be corrected in software (since you'll know the topology of each measurement), or reduced by stacking more switches in parallel. Having some resistance on the switches also prevents unintended shorts. Silver linings! Lastly, I don't trust the 12-bit converters on the RP2350[7] which manages the Jumperless connections. In the absence of an audio interface, I would prefer at least 16-bit ADCs and DACs for any critical audio work. For this purpose, I'm using my handy Analog Discovery 3, whose 14-bit converters can be supersampled for 16-bit quality.

Cost

We need to mention the cost of the device. A Jumperless v5 would set you back $349USD at the time of the crowdfunding campaign (and a little more if ordering later). If you use it to duplicate a mono $500 boutique guitar pedal, it has already paid for itself! Let's investigate how: the idea is to supersample the analog input and output of the device, inserting two 48kHz signals one after the other in the time of one 96kHz signal. This way, we can process a stereo signal through a mono analog device in real-time! Of course, it won't sound exactly the same, due to any bias shifts from capacitors charging at the new rate[8]. Still, it's a neat party trick, and showcases how out-of-the-box thinking rewards Jumperless owners with extra functionality. Another way to look at the cost is that equipment from Rohde und Schwartz costs tens of thousands of euros, and it sure won't reconfigure your circuit for you! ;) Finally, much of the cost is in the required analog crossbar switches. Kevin says: “if you think you can make it cheaper, go ahead, it's open hardware”.

Unboxing

The Jumperless comes in a quality cardboard box with a magnetic lid. In a nice touch, the open source schematics are printed in full colour on the inside of the cover. The box also contains an adapter board to Raspberry Pi[9] headers which can additionally friction fit a mini OLED screen[8], another adapter breaking out the DAC/ADC/GPIO/3V/5V, a probe that can be used[10] to make connections, a ribbon cable to the Bus Pirate[8], and of course the aforementioned card with safety instructions. The probe connects to the Jumperless over headphone jack (TRRRS for those in the know!). Notably missing is the required USB-C cable, but there are plenty of those around anyway.

Getting Started

The QR code on the box leads to a GitHub page with instructions to get you up and running with the device. After sourcing an errant USB-C cable and connecting it to the computer, I downloaded the control app. Due to a temporary packaging issue with the Mac app, I ran the provided python control app directly from the terminal. Much like the Analog Discovery 3, a large part of the value in these devices is in the software, and the Jumperless impresses in this regard. The app is cross platform (Windows/Mac/Linux) and keeps getting new features. In fact, the latest version of the firmware conveniently has a micropython REPL[11].

[7] The previous iteration (RP2040) had notoriously bad converters.
[8] You can correct the bias in software! Seeing a pattern?
[9] Not included
[10] But is not necessary, more on that later. ;)
[11] https://docs.micropython.org/en/latest/reference/repl.html , https://jumperless-docs.readthedocs.io/en/latest/09.5-micropythonAPIreference/

It's all about making connections

The instructions are a lot to take in, especially about the powerful "idle" mode. Note that breadboard "columns" on the Jumperless are called "rows" in the documentation. There are at least 4 ways to control it. First, a computer with a serial connection over the same USB-C cable that powers it. For this, you can either use the official app or a python script. I prefer the latter approach. Another way to create and delete connections is the included probe. Just touch two columns on the board to make a connection! What could be easier? Well, not having to do that of course! My goal is automated data acquisition. This leads to the third way to make connections: a micropython script running on the RP2350 that manages the Jumperless. Finally, you can use the click wheel to navigate a menu presented on LEDs on the breadboard itself. It's surprisingly readable. While it's nice to have that option, I wouldn't choose this as my main input method for making connections, as it's slow.

It does what now?!

The Jumperless can be extremely flexible. In many cases, limitations can be bypassed with clever scripting or workarounds. For example, you should be able to stack paths to handle higher voltages. I didn't try that, as I didn't want to break it on the first day, despite Kevin's insistence he'd happily replace any burnt chips (no questions asked). You can also duplicate paths in parallel to halve the crossbar resistance[12]. You can precisely control the switch timing to charge a capacitor to a predetermined bias point. It can measure a voltage or current and make connection decisions based on that.

The digital side

When not managing the connections, the RP2350 is freely programmable. In practice, this makes the Jumperless that world's most expensive RP2350 dev board! It even has RISC-V cores, if you want to experiment with that. While not as powerful as, say, the STM32H7 on a Daisy Seed board[13], the RP2350 packs quite a punch: a dual core at 150MHz which easily overclocks to 300MHz, hardware floating point[14], and PIOs for PWM waveforms.

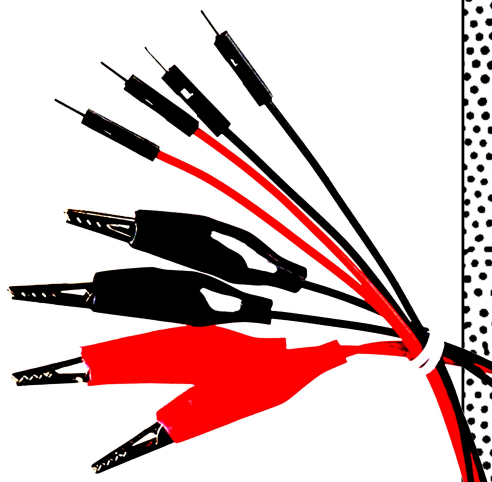
What is it?

The Jumperless defies simple categorization. Is it a modular synthesizer? A guitar pedal? An analog computer? A Field Programmable Analog Array (FPAA) ? A matrix mixer? A livecoding aid? It can be all of those things and more. And while I'm fixated on audio, I'm sure you, dear zine reader, can come up with even more ideas that use it. I'd love to hear all about it, so get in touch!

For audio effect plugins, as well as more updates on data acquisition, circuit modeling, and the future of audio, you can join my community with the following link: *https://discord.gg/me3udYDw6r*

Acknowledgments

Special thanks to Renan for help with analog electronics and everyone who helped with proofreading.



[12] In fact, the Jumperless does this by default

[13] https://electro-smith.com/products/daisy-seed

[14] -ish... Fused Multiply-Add is supported, but takes multiple cycles.

Turning Old Radios Into Guitar Amps

Ashton Felix

Ever wanted a gritty, lo-fi guitar tone without shelling out for boutique gear? Try this: turn an old portable radio into a functioning guitar amp by wiring a 1/4" jack to the volume pot. It's simple, cheap, and gives you a unique sound no commercial amp can replicate.

Why Old Radios?

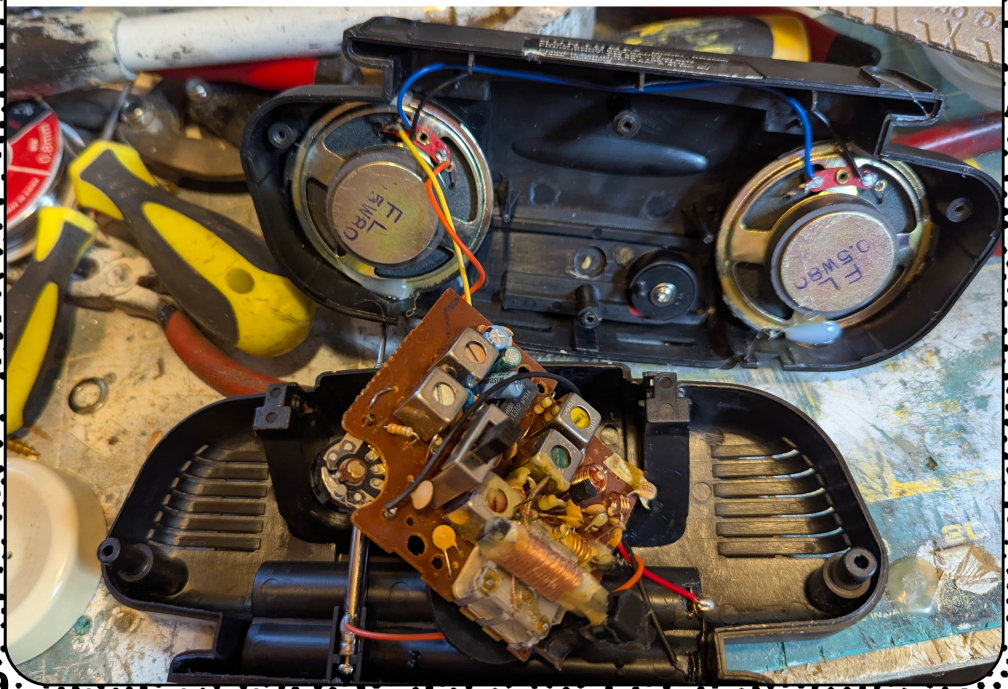
Radios from the '50s to '80s are perfect DIY candidates. They're cheap, built with analog components, and often have surprisingly decent speakers. When you bypass the tuner and feed your guitar signal directly into the amp section, the result is raw, dirty, and full of character. Perfect for garage rock, blues, or experimental tones.

What You'll Need:

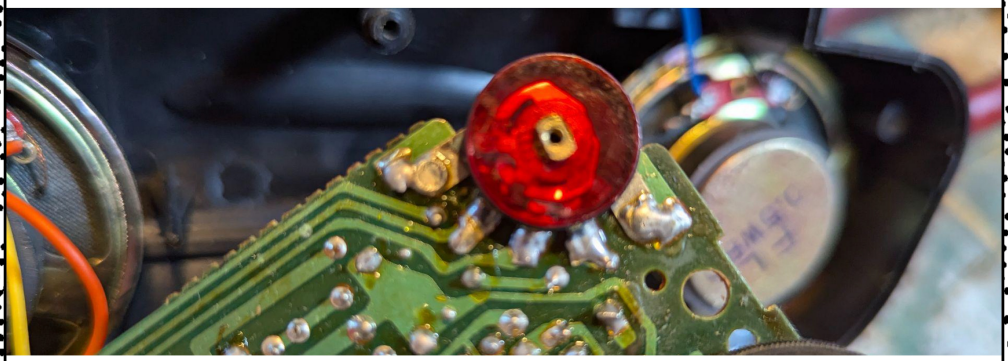
- Old portable analog radio (transistor-style is best)
- 1/4" mono input jack (preferably switching type)
- Soldering iron & solder
- Screwdriver, wire cutters, electrical tape or heat shrink
- (Optional) Multimeter and alligator clips for testing

The Quick How-To

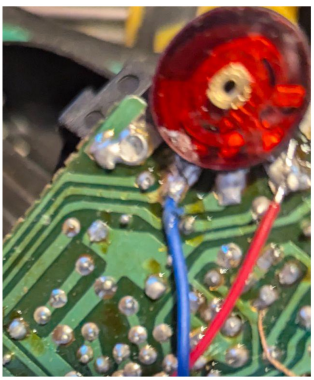
1. Unplug it, remove the batteries, and open the case.



2. Locate the volume potentiometer — it usually has three lugs: input, wiper (output), and ground.



3. Drill & Mount the Jack
4. Drill a hole in the case and install the 1/4" jack. Use a switching jack if possible — it'll cut off the tuner audio when your guitar is plugged in.
5. Wire It Up. Tip (hot) from the jack goes to the input lug on the volume pot. Sleeve (ground) connects to the pot's ground or chassis.



6. Plug in your guitar, power the radio, and turn up the volume. You should hear your guitar signal through the speaker — raw and beautiful.
7. Once everything works, close it back up, secure your wiring, and rock on.

Bonus Mods:

- Tone knob: Add a simple passive tone circuit to roll off highs.
- Better speaker: Upgrade to a fuller-range speaker if you've got the space.
- External speaker jack: For more volume or different voicings.

Final Thought

Turning a radio into a guitar amp is about more than tone — it's about creativity and using what you've got. Every radio sounds a little different, and that's the magic. So next time you spot one at a garage sale, grab it. Your next favorite amp might be hiding behind that dusty AM dial.

"Les Belles Noiseuses"[1] is both a musical project that started in 2010 and the name of the electronic instrument used in the eponymous project. I had already been experimenting with circuit bending and homemade instruments for a few years (for example, using the complex sound generator SN76477[2], which remains one of my favorite ICs) when I saw a fashion doll in a dollar store (a cheap, poor copy of a Barbie doll) that, for some reason, appealed to me. The idea that came to me was to insert a small electronic circuit inside the body, use the arms as potentiometers, and place an audio jack in the head. I started buying a few of them, aiming to create a series of cheap, relatively easy-to-replicate, unconventional, and unique instruments.

I began with the old, classic 555 IC (more precisely, the 556 IC) but was quickly limited by the lack of sound variety. I found myself caught between two seemingly irreconcilable goals: to create instruments that would be almost identical from the standpoint of electronics, operation, and general design, but that would each be sonically distinct, making each instrument unique. The ultimate goal was to perform live with these dolls and mix their sounds, essentially acting as a DJ of sorts, where the "D" stands for Doll.

Of course, using analog circuitry, one could argue that there is an infinite variety of sounds that could be created. But with limits on the circuit complexity, size, replicability, and cost, all these circuits would likely end up sounding quite similar (like oscillators with different frequency ranges).

I was already familiar with microcontrollers but had never used them for generating sounds. Using a microcontroller has a number of advantages. The complexity of the circuit as well as the number of parts are relatively low, sensors can be used through the analog-digital converters of the microcontroller (notably potentiometers) and one can easily, for example, add LEDs with fanciful control. More importantly, because the sound is ultimately generated by a program, a wide variety of sounds can potentially be generated relatively easily with the exact same circuit. Once the electronic circuit has been designed, sound generation only becomes a matter of programming.

I decided on the ATtiny85[3] microcontroller for its low cost, small size, and internal oscillator. The first thing that comes to mind when using a microcontroller for generating sound is DDS (Direct Digital Synthesis, a method of creating analog waveforms from a digital source). However, DDS soon started to feel limited too, and I was looking for new ideas. I expanded on DDS by using it with sequencer-like programs and other methods, used sound samples, or added other types of sensors (light sensors, tilt switches, even a temperature sensor) to add more variety.

---

[1] https://lbn48.bandcamp.com/

[2] https://en.wikipedia.org/wiki/Texas_Instruments_SN76477

[3] https://www.microchip.com/en-us/product/attiny85

But the true breakthrough came when I stumbled across an online post in 2012 titled "Algorithmic Symphonies from One Line of Code"[4] (there is an online version that you can try[5]). The timing was perfect, and I immediately thought that the variety and simplicity offered by these single lines of code would be a perfect fit for my project. Moreover, there was something about the kind of weird 8-bit music that could be generated that really caught my ears. Finally, it was relatively easy to implement on a microcontroller, which, by the way, is also an 8-bit microcontroller.

Not only did this algorithmic approach allowed me to create numerous unique sounds with relative ease (making the device itself is still the most complicated and time-consuming part, especially when fitting all the parts in and on the doll), but when mixed with other techniques or approaches (DDS, sequences, etc.), it gave me even more, almost infinite, possibilities. I continued to experiment with these algorithms, other techniques, and sensors and recently even started to go back to a non-microcontroller solution, using a PT2399[6] with a microphone!

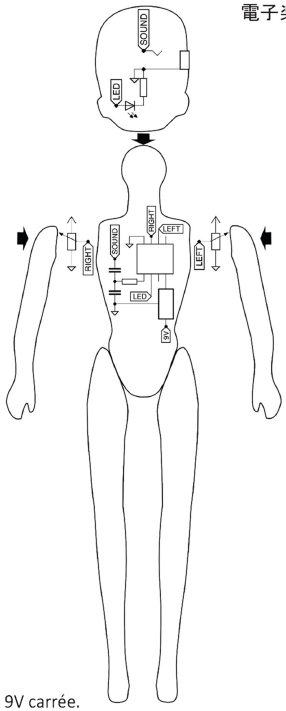
But I still love the idea and the incredible potential and variety of these simple algorithms, and I'm still making dolls that use this basic concept, nearly 15 years after the first one! I hope to soon reach the 500th "Les Belles Noiseuses" (though I still have to make almost 80 of them)!

[4] https://countercomplex.blogspot.com/2011/10/algorithmic-symphonies-from-one-line-of.html
[5] https://wurstcaptures.untergrund.net/music/
[6] https://www.electrosmash.com/pt2399-analysis

23n1.zeni@gmail.com

Instrument de musique électronique
Electronic music instrument
電子楽器

Les Belles Noiseuses



Alimentation: pile 9V carrée.
Sortie son: mono jack 6.35mm, connecteur localisé sur la tête.
Mode d'emploi: tourner les bras pour modifier le son.
Précautions:
- bien tenir le connecteur son de la poupée lors du branchement ou débranchement.
- la poupée ne possèdant ni bouton d'alimentation ni contrôle de volume, faire attention au volume lors du branchement à des hauts-parleurs ou à un casque audio.
- débrancher la pile après utilisation.
- faire très attention à la polarité en branchant la pile.

Power supply: 9V square battery.
Sound output: 1/4 inch mono jack , connector located on the head.
How to use: rotate the arms to modify the sound.
Caution:
- hold the sound connector of the doll tightly when plugging or unplugging.
- as the doll features no power switch nor volume control, be careful to volume when plugging it to speakers or headphones.
- unplug the battery after use.
- be very careful about the polarity of the battery when plugging it.

電源: 電池006P/9V.
音出力: モノラルジャック（後頭部）
使い方: 腕を回すと、音が変わります。
ご使用上の注意:
＊ 出力コードを抜き差しするときには、コネクターをしっかり押さえてください。
＊ 人形には電源スイッチや音量調節がないため、使用する際にはスピーカーやヘッドフォンのボリュームに気をつけてください。また電池消耗を防ぐため、使用後はすぐに電池をはずしましょう。
＊ 電池を取り付ける際に、±を間違えないようにお気をつけください。破損の恐れがあります。

VCO FREAK MACHINES

by 2233mar

I wanted to write this article to give beginners something that's easy to make, fun to listen to and play with, and *not* an Atari Punk Console. Don't get me wrong- I love that thing. But let's instead have some fun with one of the most basic building blocks of synthesizers: the Voltage Controlled Oscillator. VCO's open the door to AM/FM synthesis, synth fuzz, and tons of other useful effects and sound design techniques. It's recommended to have some prerequisite knowledge of Ohm's law, voltage dividers, and maybe coupling capacitors.

Intro To VCOs

VCO's come in many shapes and sizes, and are usually pretty easy to wire. If you want to stay on the dirt cheap side of things, you can whip one up with a 555-timer chip. Another option, which I prefer, is the CD4046 (fig. 1). It provides a more even pulse width and a larger frequency range. The frequency range is set by a resistor and a capacitor, which in this case is 10nF and 22kΩ. Lowering either value raises the maximum frequency. Adding a simple voltage divider to the voltage input pin of the VCO allows us to sweep the frequency range and get a basic square wave output.

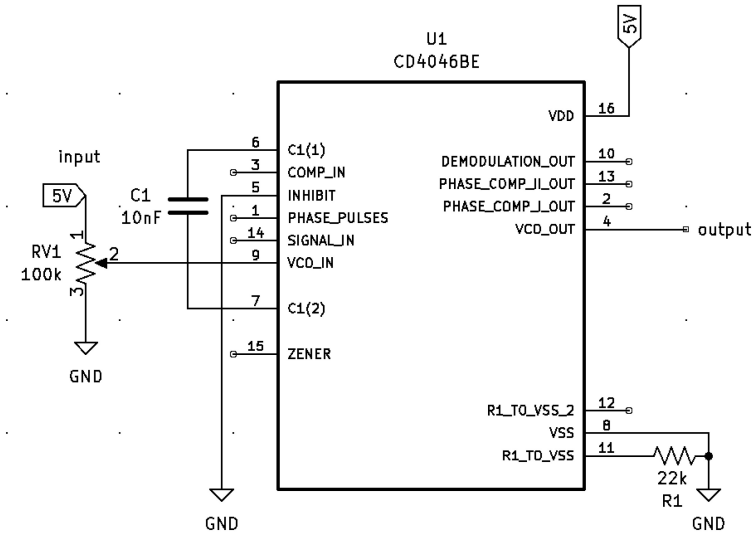


Fig. 1

The Build

Things start getting freaky when you plug a VCO into another VCO. This is the most basic form of AM/FM synthesis. For the sake of simplicity, we will only focus on the modulation signal. Fig. 2 shows the complete circuit. The voltage divider outlined in red sets the frequency

of our modulating oscillator. The voltage divider outlined in green attenuates this signal, functioning as a modulation depth control. Finally we put this signal into another VCO and listen to its output. (You can extend this ladder of VCOs with as many CD4046's as you like!) Because we are ignoring the carrier frequency, the low half-cycle of our modulation signal causes the carrier oscillator to go to 0hz, which is silent. The result is a kind of chop tremolo of our original oscillator- more similar to AM than FM, but still freaky. For an FM sound, you can bias the modulating signal to an additional voltage divider control.

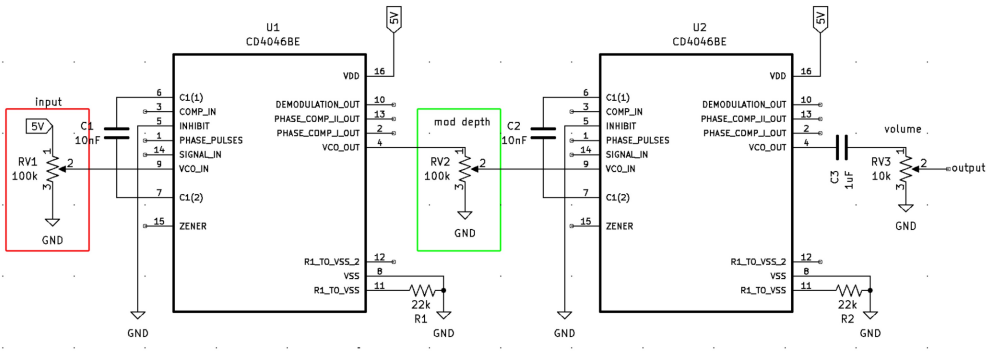


Fig. 2

Bonus: Light Mods

For extra fun, you can create a voltage divider using a photocell (a.k.a light dependent resistor or vactrol), and use it for any of the voltage dividers in this circuit (fig. 3). Just make sure that the resistor value is somewhere in the middle of the photocell's resistance range, which 10kΩ commonly is. Now you can control these parameters of your circuit by shining a light or waving your hand over your circuit.

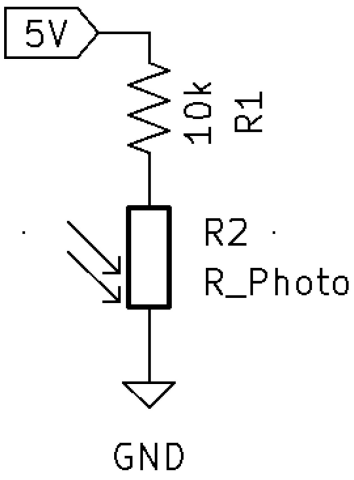


Fig. 3

ALL-PASS FILTERS

Simon Weis
simon@darkpalace.studio

For audio people, filters are ubiquitous. Terms like "low-cut filter," "bell peak," or "low shelf" are familiar to most mixers, producers, and sound designers, and they regularly use those tools to improve their sounds and mixes. All those filters have in common that they alter the volume of specific frequencies thereby enabling creative sound shaping.

Filtering Examples

- When a recording has a particularly strong low end, a low-cut filter can be used to reduce the excess bass below the filter frequency.
- If a vocal recording lacks clarity, a boost at 2.5 kHz using a bell filter can improve the perception of the text, especially in a dense mix.
- High-gain distorted electric guitars sound even nastier with a notch filter around 1 kHz.
- Using shelf EQs to add "boom and fizz" with analog devices like a pultec EQ.

What are these “All-pass Filters”?

Much less known than the filters mentioned above are "all-pass filters." They are named that because they pass all frequencies equally, thus producing no change in overall volume. The purpose and usefulness of these all-pass filters may seem questionable at first, since (as just explained) they do not apparently alter the sound. The trick is that filters not only change the volume of specific frequencies, but also their phase. This is the fundamental concept behind any filter (except for weird digital filters that I will not talk about here).

The characteristic of an all-pass filter is this: While an all-pass has a flat frequency response, it introduces a frequency-dependent phase shift. This phase shift can be seen as a frequency-dependent delay, where higher frequencies experience increasing delay.

Figure 1 shows the volume (red) and phase (blue) of a first order all-pass filter. Figure 2 shows that of a second order all-pass filter.

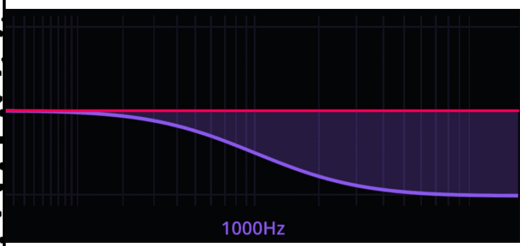


Fig. 1

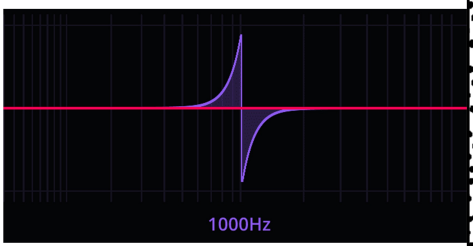


Fig. 2

All-pass as the foundation of all filters

All-pass filters are the basic building block for building all other filters. Furthermore, all-pass filters offer a wide range of sound design possibilities. that we will discuss in the next sections.

- A high-cut filter is the original signal and the signal from a first order all-pass filter summed together. As just explained, the low frequencies stay unchanged by the all-pass, but higher frequencies experience a phase shift of up to 180°. Adding the processed signal to the original causes the higher frequencies to cancel out, while the lower frequencies simply add up.

- Conversely, one can build a low-cut filter from a first-order all-pass filter by subtracting the all-pass signal from the original signal.
- Other filter types like shelves, bells, notches and bandpass filters can be created in similar ways.

Delays and Reverbs

When looking at reverbs, one critical aspect is the diffusion of the reverb tail. Ideally, the reverb tail is smudged without adding much coloration or timbre of the process itself. All-pass filters are a perfect match for this task. They delay certain parts of the signal and have a flat frequency response. Combining all-pass filters with a delay network is the basic concept behind how algorithmic reverbs are generally built. Common implementations used in this context are Schröder all-pass[1] and Gerzon all-pass[2].

Stacking All-pass Filters

Stacking multiple all-pass filters can yield super interesting results. Similar to the application in reverbs above, this will smear out the signal.

Let's take a look at a snare drum hit. This is a transient heavy sound with a certain amount of decay to it. Depending on the tuning of the drum you get a resonance frequency and some overtones (or more, if the snare wires are engaged).

By applying a larger number of all-pass filters to it, you can :

- disperse the transient by a certain amount, leading to a more equal distribution of energy over time
- make the snare's resonance frequency ring out significantly longer by tuning up the all-pass filters to that frequency and bringing them close to self-oscillation
- stretching each snare hit into a crazy pitched laser sound by applying several all-pass filters in the 150-400 Hz range at increasing frequencies.

This opens up a wide range of sound design possibilities, from a slight reduction of transients to a pronounced "laser sound", depending on the number of all-pass filters and their individual frequencies.

The Chameleon All-pass Filter plugin suite

Unfortunately, even in 2025, many DAWs still ship without all-pass filters. There are existing plugins available, but they either have a hefty price tag, a hard to use UI, or lack configurability and performance (stacking up many filters can make your CPU go very hot).

That's what led me to develop and release the Chameleon all-pass filter suite. It is a comprehensive package that not only provides access to basic all-pass filters of various orders (Chameleon 1 and Chameleon 2), but also two more advanced plugins that enable deep sound design options (Chameleon N and Chameleon X).

Additionally Chameleon features include:

- modern UI
- midi keytracking
- oversampling
- a crazy per-filter clipping option
- DRM free
- reasonably priced

Chameleon is provided as VST3, CLAP, AU and LV2 on Windows, macOS (intel and apple silicon) and Linux. Check it out at: **darkpalace.studio/products/chameleon/**

---

[1] https://ccrma.stanford.edu/~jos/pasp/Allpass_Two_Combs.html
[2] https://ccrma.stanford.edu/~jos/pasp/Gerzon_Nested_MIMO_Allpass.html

BODYBEND

María Takadanobaba
takadanobabapinipona@gmail.com takadanobaba.bandcamp.com IG: pinitan.takadanobaba

Rostro Bending is a circuit-bent sound sculpture and live performance, where glitch, noise, and hacked electronics intersect with hyper-realistic sculpture and autobiographical material. At its center is a silicone replica of my human face, embedded with two distinct electronic systems: one built from obsolete toys with circuit-bent modifications, and another based on a DIY Atari Punk Console, generating lo-fi 8-bit glitches and noise textures in real time.

The process behind Rostro Bending is deeply rooted in self-taught experimentation. Over the course of a year, I collected discarded electronic toys from the streets, disassembled and restored them, and explored their circuits using trial-and-error methods with alligator clips, a soldering iron, and my magic and steady hand. Most of the toys I found were modern and compact, with surface-mounted components (SMD) and integrated circuits, making traditional circuit bending difficult due to the lack of visible resistors or accessible traces. In contrast, older toys allow for clearer navigation of the circuit's logic: visible capacitors, transistors, and resistors invite creative sabotage—tweaking pitch, rhythm, or triggering unexpected sonic feedback. To overcome the constraints of miniaturized modern circuits, I decided to hybridize her system by integrating a DIY Atari Punk console, introducing a stable source of noisy, glitchy tones with analog tactility. Two glitch buttons, a set of pre-programmed melodies, and two speakers of different sizes were embedded into the sculpture to create an unpredictable, interactive, and layered audio output.

The sculpture itself—hyper-realistic, eerily human—is not merely a support structure but part of the conceptual and sonic ecosystem. The process of copying my own face involved destruction and reconstruction: breaking down the original, remolding, adding new textures, and casting in silicone. That same logic applies to the sound-making components: dismantle, intervene, glitch, resurrect.

In live performance, the piece becomes an expressive interface. I manipulate the facial circuits through touch, gesture, and movement, letting the hacked circuits "speak" in dissonant, noisy, fragile tones. Through Rostro Bending, I can explore the body as both a site of memory and mutation, where my personal identity collides with technological detritus.

Rostro Bending is the first piece in a larger body of work is titled BODYBEND—a multidisciplinary, long-term project where different parts of my body are replicated in hyper-realistic silicone sculptures: torso, arm, hands, and a separate sculptural microphone piece shaped like a penis, anus, and breast. Each body part is embedded with circuit-bent electronics, hacked by hand from discarded toys and DIY synthesizers. The result is a series of interactive sound-sculptures that explore the blurred boundaries between flesh and machine, intimacy and technology, organic and synthetic.

These "machine-bodies" do not seek to imitate life, but rather to confront it—to present the post-human as something wounded, fragmented, and open to reinvention. Through sound, touch, and physical presence, BODYBEND creates a hybrid performative language that resists conventional representations of the body and embraces error, noise, and transformation as sources of power.

MARÍA TAKADANOBABA

POSTROBENDINO

ROSTRO SINTÉTICO

INTERVIEW: MORITZ KLEIN

https://youtube.com/@MoritzKlein0

Moritz Klein is a music artist, circuit designer, and online educator based in Berlin. His YouTube videos dive deep into the inner workings of synthesizers, offering clear explanations and examples of useful circuits. His channel (under the same name) has amassed over 130,000 subscribers since its inception in 2020. Additionally, Moritz has partnered with Erica Synths to create several DIY synthesizer kits as well as a full-fledged prototyping station called Labor, all available for purchase at ericasynths.lv. We wanted to ask him about his journey in online education, his partnership with Erica Synths, and some of his technical insights. In early August 2025, he kindly agreed to an interview with Circuit Punk for this summer issue.

Where are you from and how did you first get into music electronics?

So I'm from a city called Paderborn in Germany, or more specifically a little village close to that town. I started playing drums when I was 11, and started playing in a band when I was 17. I did that for a couple years, we played some shows, and it was really fun. Then a bit later I moved to Berlin where I live now, and I expected to find people to make music with, but the city is very much fixated on electronic music and clubbing and stuff, so it was difficult to find people who wanted to play in a band. At some point I just got frustrated and started looking into making electronic music. I had been interested in synthesizers for a while but I never really owned any, and then I started buying them. One that I bought was the MS-20 Mini, which is a semi-modular synth, and that was very intriguing to me that you could just plug in cables and the sound changes. I couldn't wrap my head around it but I was intrigued by it. I remember stumbling on a Look Mum No Computer video on how to build an oscillator and that set me on this path. I went out and bought a couple chips and a breadboard and started playing around. I basically then learned everything just by trial and error. I was writing my master's thesis at the time but was studying cultural studies, so it was not related. This was one of those things where I got really excited about it and got up very early in the morning to just sit at my desk and build something. It's been a while since I was as excited about something as I was about this, and it just progressed from there.

You mentioned Look Mum No Computer. Are there any other teachers, books or websites that really helped you along the way or that you'd recommend?

So yeah, there's one that was really helpful. It's a series called Logic Noise, it's a series on hackaday.com. That's really good, but that's the only resource I could find that was written in a way that appealed to me. Cause I'm not a studied engineer and many of the tutorials I could find were written with engineering jargon- I felt like it wasn't really outcome focused. Oftentimes these engineering discussions are very nit-picky, they delve into the details and all the edge cases and whatever. But for me, I was looking for something that was like, 'Hey if you want to build an oscillator, try this circuit. And here's why it works.' And that series was the only one I could find that did that.

When analyzing schematics, such as the Roland T909 in your video, "This analog circuit can sound like any drum", how do you break down something so large and complex into more digestible pieces? Is it familiarity?

Only partially. The thing with the Roland schematics is they often do very unorthodox things, and they use very basic parts in ways you don't see anywhere else, so oftentimes it's puzzling to look at. The way that I approached it was first of all isolating the voices, cause it's one huge rat's nest of things. One thing you can assume with that stuff is that it's always gonna be subtractive. So you can always look for an oscillator or some kind of sound source and then just work your way through it from there. That's basically what I did. In the case of the kick, start with the oscillator; identify it, isolate it, build it on a breadboard, and then try to understand how it works and where the signal goes after. So in the case of the kick, it's a triangle oscillator that goes into a wave shaper that turns the triangle into a sine wave. Then that goes into a VCA. And the VCA on the 909 is really bizarre, so then I needed to understand 'How does this work?' and there are all kinds of restrictions that come out of the way the VCA works. I try to start with the simplest element, and then it just sprawls out from there.

Are there any components or building blocks that are underappreciated in your opinion?

One thing that I really like is diodes, just plain old diodes. And it's because usually when people are taught diodes, they're taught that it's a one-way street for electricity and that it's on or off. It's like, either it blocks or it conducts. But [diodes] have this fun characteristic that they can be opened to varying degrees. So if you increase the input voltage, they slowly open up. And you can abuse that in various ways to make voltage-controlled resistors, which is extremely helpful in a synth context. And I feel that it's not clear to a lot of people that that's possible. I just made a short video on a VCA using just diodes and a resistor. What's interesting there is people have this very quick reaction when they look at the circuit, and they're like, 'Oh, this is a limiter.' Because to them, it's like, 'Oh, diodes are on or off. So this cannot attenuate. This must limit.' And I get where they're coming from. But yeah, since you asked for "underappreciated" components, I think I would have to go with diodes.

Your series with Erica Synths has taught many people how to build and design music gear. What has it been like to design these kits, and how did you think about their educational use?

This at first was a really tricky one because…Erica Synths approached me in 2021. They sent me an email and said they wanted to reboot their DIY kits series because the DIY kits they had previously were very complex, and they had trouble with customer support because people would send in defective builds all the time. They wanted to start fresh and do something simpler. They saw my videos and were like, 'Hey, this could work.' For me, it was kind of bizarre because I obviously knew about them and thought they were doing really good stuff, and I never thought I'd collaborate with them, and they just sent an email, which is crazy. At first, I declined because, for one, I didn't feel up to the task—I didn't feel like I could produce something that was ready to be used commercially. But also, I had this weird pride at the time where I was like, 'I don't want to monetize my work.' I didn't run ads on my videos or anything. I think I was just very averse to the whole idea of making a product. They came back and were like 'Please reconsider, this is a really good opportunity,' And at some point I was like, 'Ok, I'm gonna try it.' We agreed on releasing them on a very tight schedule of one every 4-6 weeks. I was totally naive, like, 'That's gonna be fine. Easy.' But yeah, preparing a circuit, doing the prototype, assembling the prototype, testing it, doing revisions, writing a manual, and then releasing it…that was a crazy time. That was in 2022. I just didn't have a life at all. I was just cranking out stuff all the time, but I also had a full-time job, and balancing that was insane.

As for the educational value of the kit, it was interesting because that wasn't really spelled out back then. Erica Synths' focus was like, 'We want to have a kit that people can assemble.' To them, it was just about soldering. In my experience, most of the mainstream DIY

stuff is about that. For example, there's this fair called Super Booth here in Berlin, and I do workshops there regularly. And they are totally not set up for breadboarding. The room that they provide is just for soldering. It's also a huge kind of classroom. It's super loud. There's no whiteboards. So to them, DIY is soldering. And for Erica Synths, it was the same. I felt that's what's going to set it apart is this breadboarding thing, and this idea that you can learn about the circuit as you're breadboarding it. But it really felt like I had to fight and push for that, and also to highlight myself in the marketing and make sure that people get that that's the idea. And it's kind of still an issue. Now we have the Labor, this prototyping station, and I feel like with that it's become clearer to people what the series is about. Because the thing is basically a breadboard. But yeah, it's still challenging. And I still have some, not regrets, but things I would like to do differently in the future. For example, right now, the DIY kits don't come with a printed manual. So you just get the box, the components, and the PCB. So it's very much like you open it and you feel like, 'Okay, this is meant to be soldered together.' And for a long time, I've been wanting to think about how we can package this differently, and how we can present this in a way that when you get it, it's clear that this is meant to be breadboarded first and then soldered. So yeah, long-winded answer. But yeah, the educational thing is a challenge because it really feels like it's kind of uncharted territory. So I need to always make sure that people understand what the educational value is.

Do you have any strong feelings about digital versus analog, and do you think the Moritz Klein channel will ever teach something more in the digital realm?

So far, I have not done anything with digital. It's always hard to draw the line between analog and digital, because something like a shift register, for example, by definition is digital. If you mean coding, then what I'm kind of averse to is using a microcontroller for something that you can do with analog components. Because to me, what I enjoy about making circuits is how close to the metal you can be. Because I'm a software developer by day, trade, and so I really prefer to be super close to the metal, and then to be able to see how I can manipulate real signals and not just representations of signals. That, plus, I feel like once you start using a microcontroller, you start kind of taking shortcuts in a way to some degree. It's often like, I don't know what the English equivalent is, but we have a German saying which translates to "shooting sparrows with a cannon." You don't need this insanely complex piece of engineering to make an envelope, for example. So it's a little bit out of balance. That said, I would be interested in using digital tech when it's kind of a use case that really calls for it. Way back, for example, I did a quantizer, which in essence is digital, because it took an analog voltage and then ran it through an analog-to-digital converter. Then I ran that through a bunch of logic gates to kind of see if the voltage is kind of inside the major scale. And if it wasn't, then I would correct it in the digital realm, and then I would feed it into a digital-to-analog converter and then have the output. So for those use cases, for example, it's just a no-brainer, because you can't really do that in an analog way.

Get Moritz Klein x Erica Synths DIY kits: ericasynths.lv/shop/diy-kits-1/
Subscribe to Moritz Klein's YouTube channel: youtube.com/@MoritzKlein0
Check out exclusive content on Patreon: patreon.com/moritzklein

SIMULATION-ASSISTED DISTILLATION OF A CIRCUIT'S ESSENCE

dylan159
bentfishbowl.wixsite.com/electronics

Many of the earlier circuits I shared on Freestompboxes and on my blog were adaptations of popular effects pedals in which I tried to break down and simplify them as much as possible without compromising their characteristic timbre. This involved reducing the number of components, using standard values, like those in the E6 series most of the time, forgoing exotic parts, and why not, applying the principles of low noise design (for which I recommend Self's Small Signal Audio Design as reference) to even surpass the original in this rare case where we can objectively say what is better. This last goal usually involves keeping resistors small and avoiding attenuation before amplification wherever possible[1].

This is a useful exercise both for the performer, who gains a better understanding of the circuit, and for others, who get a circuit which is easier to build and just as good, if not better. That's why I wanted to share some tips about my process so that you can apply it to the many other circuits I haven't tackled myself.

Planning

The first step is a preliminary analysis of the circuit, and identification of functional blocks. Some experience helps here, and a first glance can already tell some ways in which the circuit can be modified, based on what you know from other circuits. There is no good substitute for the time spent perusing schematics of BOSS and the likes, but don't worry: that will come with time and what we will discuss later is just as effective.

The second point deserves some words: audio circuits are mostly serial, so it is a very effective strategy to break them down into blocks, leaving convoluted feedback networks for later. A good starting point is checking Electrosmash, or my own analyses, and also having some basic electronics theory. Usually, one active component defines its own stage, but we can also have completely passive stages that can stand on their own (think of some filters), or sometimes amplifiers so tightly knotted in feedback that it makes little sense to separate them. It's important to remember that even when neatly separated into a series, these blocks can interact with each other, and we will see how we can account for that.

Simulating AC

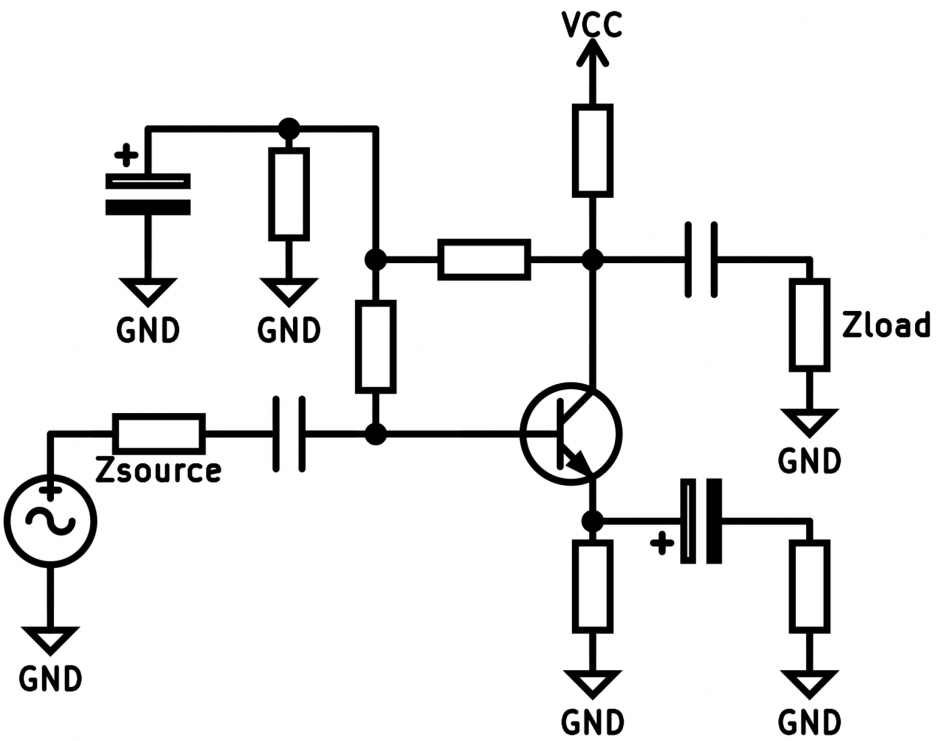
Circuit simulators can easily take whole circuits at once, so do we need to bother sectioning them into blocks first? I think so! If we are planning an intervention, it might be hard to see differences when probing a whole circuit or point down why things don't turn out as we wanted, but when it all falls apart is with nonlinearity, a common presence in our favourite circuits. Yes, you really need to simulate that clipping stage separately.

This is especially true when using my favourite tool, small-signal AC analysis (.ac command in LTspice), which completely ignores nonlinearities. This kind of analysis is the core

[1] You should really read the first part of that book and Art of Electronics, though.

of my process, because it lets you plot gain and frequency responses for the whole audio spectrum, probably the most important aspects of any audio circuit, as well as input and output impedances. For a technical tip, 20 to 20kHz with 100-1,000 points per octave/decade are usually enough and keep the simulation time low.

This is when we can start to experiment with cutting and merging components, approximating one value and compensating with another, and scaling impedances up and down! A word of warning, and one take-home message though: we still want to account for the loading effects of the current stage on its source, and of the following on the current stage. This is best explained with the following schematic (one of my favourite common emitter topologies):



with most stages, we need to account for source and load impedances

While we like to work with ideal voltage sources and the simulation works even without a load, they can affect the output of the block, sometimes negligibly, sometimes drastically! Usually resistors are enough (even a transistor amplifier can be modelled as a resistor if it's not saturating), but you might want to use more complicated networks (as for a guitar pickup), or even parts of other blocks if necessary. Those "dummy" impedances are a great tool for simulation, but can also give you ideas on what you can simplify, since they are not restricted to the block's input or output: you can swap in a Thévenin equivalent (which sound complicated but it really isn't) wherever you have a voltage divider for example, if you want to focus on its impedance.

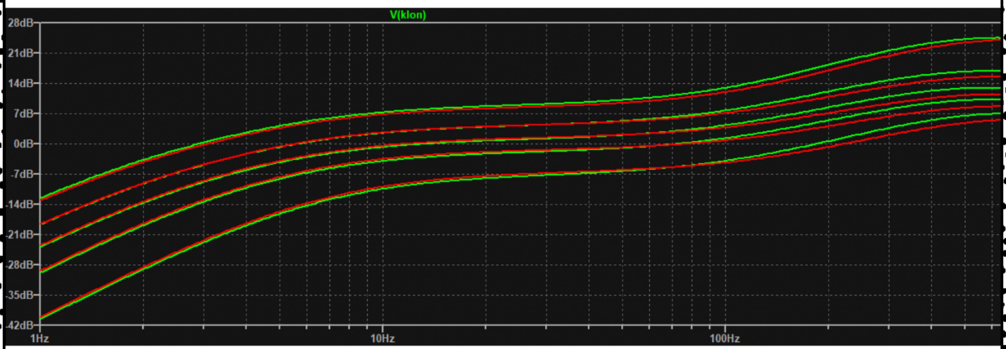
Simulating DC

Even when we only care about our AC audio, we often still want to worry about DC. You can use transient analysis (.tran in LTspice) to estimate clipping, under very synthetic conditions like a steady-state sine wave, but it is most useful to check DC bias conditions with no input signal. This is necessary, for example, with many simple transistor circuits, where simulation can give you a more reliable and quicker answer than pen-and-paper. Of course, your transistor is never going to be the same as the simulation model, so it is a good idea to check how sensitive the circuit is to perturbations of the supply, resistor values and different part numbers. Transient analysis is also useful for oscillators, of course, and you might have to add some parameters and change the length from a default 10 ms in some cases (slow waveforms, dealing with charging capacitors, etc.).

Exploring the parameter space

The third and final tool I want to talk about is stepping. You often want to try multiple values at once, or you might not even have an idea on what order of magnitude the value should be. Maybe you just have a potentiometer or a switch and need to check what happens at several settings. In all those cases, stepping (.step in LTspice) is the solution, allowing you to sweep one value or many at once over a range, a list of candidates, and even simulate potentiometer tapers with logarithmic steps. Some software allows you to step multiple parameters at once, but focusing on one at a time is surprisingly effective and allows you to quickly pin down pivotal values, which of them don't really matter, and which are best left as they are.

You can complement this with two other techniques, assigning symbolic values to conveniently change many values at once even manually, and copying whole blocks to compare different versions one against the other (the original vs. yours, or multiple candidates). A single ideal voltage source can drive all circuits at once with no change, but this might not be true in other cases.



For example, the gain stage of the Klon (green) and of the Clown (red) at different settings

Go on and try it yourself!

That covers the basics of what you need to reinterpret popular circuits the same way I did. There are hundreds still waiting out there, and I'm excited for you to share with the world your own application of this approach and principles, all the while having fun learning electronics the same way I do!

INTERVIEW: THE AUDIO PR{}GRAMMER

theaudioprogrammer.com/

Joshua Hodge is the founder of The Audio Programmer. TAP is a website, YouTube channel, Discord community, book publisher, and organization for all things audio programming. TAP also works to connect individuals in the audio industry. When it comes to DSP and building plugins, they're ready to help with any skill level. In May, Joshua agreed to an interview to share a bit about The Audio Programmer's mission with Circuit Punk.

How did you get into audio software development, and how did it turn into The Audio Programmer?
I originally came from a DJ and a music production background. And in 2015, I decided to go to university to a program in London called Music Computing. I'd never written a line of code or done any type of programming before that and that just immersed me into this new world of creative coding and technology that, especially at that particular time, nobody was really exposed to. Very few people actually knew what happened behind the scenes of these music technology companies, and I was so fascinated by it. And during my second year at university, I decided to start The Audio Programmer as a YouTube page, more to teach myself how to code, because I was getting some resources from the university, but I knew that it wasn't going to be enough to get me hired as a developer. So I actually started it originally as a way for me to discipline myself to learn something new each week through teaching other people, and it grew from there.

For beginners interested in audio development, do you think it's necessary to have a little background in coding beforehand, or do you think audio development could be a way to learn how to code?
I think it's a little bit of both, but I would say that the best way to learn in general is to find a project that you would like to develop and then just work towards that. There are a lot of people that want to dive into the textbooks and dive into knowing how to do everything properly before they try to get into a real project. And I've found for myself, and I've seen it in many other people, that the best way to learn is just by doing and just by making something, you know? I think the main thing there is that you wanna keep the projects achievable, make sure that they're realistic. But doing practical things is the way to code, and you'll definitely learn along the way.

What programming languages or platforms are most commonly used to develop audio plug-ins? And does The Audio Programmer have any favorites?
For audio programming for plug-ins, you've gotta mainly use C++ for those. There are some other ways. Max MSP has a tool called RNBO that allows you to create plug-ins from their code. But mainly C++. In terms of tools, a lot of people use a framework called JUCE. That contains a lot of the tools and a lot of the building blocks that people would typically want to use

to create plugins- things like loading files, audio streaming, things like that. But, there are definitely some others out there, and now, what we're seeing is there are a lot of open source tools that are happening as well. So you have iPlug2 which is another plug in framework. There's also the CLAP API. So CLAP is an audio plugin, format, you know, like VST or, like, AU, but it's open source. And so there are many people that are using CLAP to develop their plug-ins. CLAP has wrappers where you can take a CLAP plug in and turn it into a VST or an AU. So what they're doing is using CLAP as their primary format and then using the wrappers to convert into other plug-in formats. That's something that has been a trend lately.

And with these more recent things that you're mentioning, does it give you the same level of, sort of low level control that C++ would give you?
Yeah. So those other tools that I mentioned are all C++ tools. What I would say there is that a fully fledged framework like JUCE gives you a lot of building blocks and a lot of helpers to do things more conveniently and more quickly than you would do if you were using, let's say, the CLAP API directly. You would have to build a lot of those tools directly yourself and, you know, build those from scratch or use other open source libraries in order to do those things.

On The Audio Programmer's YouTube channel, you post lots of original content on there as well as these "digital monthly meetups." What kinds of topics and people show up at these meetups?
Yeah, it's really all things audio programming. When I started the channel, there was very little awareness on what happens in the industry. There were very few developers that were actually on social media or on any of these platforms like LinkedIn or Twitter. So the first intention was really to show people, well, here's what it's like to work in the industry, and here's the level of expectation in terms of, if you wanted to go and work for Ableton, here it is from a developer at Ableton that'll tell you exactly what it's like to work there, exactly what types of things that you should know, what type of skill set that you should have, and so on. And one of the things was just telling the stories of people in the industry. We also had the CEO of Ableton, Gerhard Behles, who came to talk about his vision for the company and talk about how Ableton started from just an idea, Max MSP, and it built into this 400+ person company. It's really about anybody who has a story, a product, an idea to share within, within the industry. So anybody's welcome from a CEO of a large music tech company down to university students who have just created their first audio plug in, and they wanna showcase those ideas and showcase what they can do.

Within your Discord community, what kinds of projects and stuff go on in there?
I'd say that there are a lot of companies that I know about and companies that I don't know about that have been built from the collaborations that have happened in the Discord. The thing that I love about it is that you have this amazing mix of people who are just starting out that may not have even written their first lines of code, all the way up to people who have experience at some of the biggest companies in the world. I've even seen developers from Apple, Meta, companies of that stature, that have been in the Discord. And I love that mixture of people just starting out with people who have experience with this and just watching the cross pollination of those. I think that that's really exciting.

I wanted to ask you a bit about machine learning stuff. The Audio Programmer channel has many videos now touching on the possibilities of machine learning and audio software development. What kinds of things do neural networks enable us to do that would otherwise be impossible?
I think the main thing that machine learning and AI has the ability to do is to take those things that are difficult to quantify within code and difficult to describe, and we can just demonstrate those things for the machine to be able to learn or be able to see what the desired outcome is. I

think that's the main thing that it offers. Obviously, there have been some really amazing things that have come out through machine learning and through AI over the past couple years. But, yeah, I think that's the main thing is that there are things that are difficult to to really just describe to a computer in traditional code, and those, machine learning algorithms help us do that.

With the machine learning technology booming so much over the last few years, there are relatively few plug-ins using it on the market, and I was curious if you had any thoughts about why that is or if that will change in the near future.
I think one of the main reasons is because a lot of developers start from machine learning as a solution before they've even really thought about the problem. I think if we have a pattern that I wish would change in the industry as a whole, I wish it would be that we are thinking about what practical problems we're solving for the user before we run to machine learning as a tool. It's like us running around with a hammer and there are a lot of screws around- we're trying to use the wrong tool for the job. So, I think what we need to do is switch around our thinking and go through our process of how we solve a problem. And then machine learning becomes part of that toolset rather than the buzzword or the thing that's on trend and just trying to see, “Well, how can we fit that into a user's workflow?”

For audio software developers that are using machine learning, what responsibilities, if any, do you think they have when it comes to ethical data sourcing and the impact on climate change, which has been talked about a lot recently?
I think that it's definitely important to take into consideration. You know, we have to protect each other. We have to protect our art. I think that when these technologies become possible, if they do become massively more convenient for the user, then, unfortunately, what happens is that the environmental impact tends to become a secondary thing, and I think that that's very unfortunate. But I think that as developers, we need to be at least made more aware of the implications of the technologies that we're building and to take those into consideration.

What projects are in the future for The Audio Programmer?
We have some partnerships that I can't quite reveal yet, but those will be making themselves known over the next few weeks. We are also reinvigorating the way that we're teaching audio programming, as well. We're making a slight slight tweak to how we do that, and I think that that's going to be exciting because there are gonna be a lot of areas covered that we haven't covered before in the past on the channel. Other than that, we're working on a lot of different projects, working with a lot of different partners, and excited for the future.

YouTube: youtube.com/@TheAudioProgrammer
Website: theaudioprogrammer.com
Discord: theaudioprogrammer.com/discord

SHOVING A PEDAL INTO A EURORACK

by Andrew Wardlaw

The Mission

There's a wonderful article floating around the web about how to turn a $5 karaoke FX card into a eurorack delay[1]. It's not a perfect delay module, but it's good enough to be useful. It also made me wonder, what other inappropriate circuits can I force into my eurorack?

I set my sights on a cheap Behringer Guitar Pedal. Now, some of this I have already worked out: I've designed a breakout module that lets a pedal circuit play nice with Eurorack, and I've used it with the PCBs of various pedal kits. But a factory pedal is a different kind of beast. They are designed to be assembled by robots, and not designed to be taken apart. And Behringer is famous for using cheap parts that may not tolerate being reworked. This could be fiddly and prone to failure. But on the other hand, maybe it will be easy, and perhaps I can even sell it for a profit? Or perhaps it'll be troublesome enough to make a good article in a zine.

My Tools

Tools can be the difference between a smooth job and throwing a PCB in frustration. I do not have professional tools. I do have what I think is a decent hobbyist bench: A decent iron (Hakko 888D) with an assortment of cheap tips, a cheap iron (my first iron, which I keep on hand for special occasions), a cheap hot-air rework station, a cheap solder sucker, wick, and pokers/tweezers/holder, etc. I do not have a heated solder sucker, or anything that costs more than $110.

The Pedal to Eurorack Adapter

This circuit (fig. 1) is simple enough that someone like me can design it. The board attenuates the incoming synth signal to guitar pedal levels, sends that signal to the pedal PCB, and then takes the output of the pedal and boosts it back to Eurorack levels. You replace the pots on the pedal PCB with hook-up wire to the adapter, and then install matching pots to the adapter. It also takes the +12v of your eurorack, knocks it down to 9v, and powers the pedal PCB.

And in the Other Corner: The Behringer UC200

Ah, Behringer, the company with a reputation for using the cheapest possible parts to make clones of other companies' designs. Once, they got mad at someone who gave them bad reviews, so they trademarked the guy's name (!) and made a fake product to make fun of him. That crossed an ethical boundary for me, and I swore I'd never buy anything from them. Apparently, I'm corruptible.

The UC200 is allegedly a clone of a Boss Super Chorus. I found schematics for both pedals online, and they seem to confirm it. According to the Sweetwater website, the UC200 is an amazing deal. Pretty photos. Glowing reviews. One of the features is a "First-class electronic on/off switch for highest signal integrity in bypass mode," which gives me hope that a specific part of the conversion process will be easy. The $25 pedal costs me $36 after tax and shipping, and Sweetwater also sends me some candy, which is always nice.

---

1 https://web.archive.org/web/20230409183627/https://store-w05iciwvaa.mybigcommerce.com/content/build_guides/Waveform_Delay_build_guide.pdf

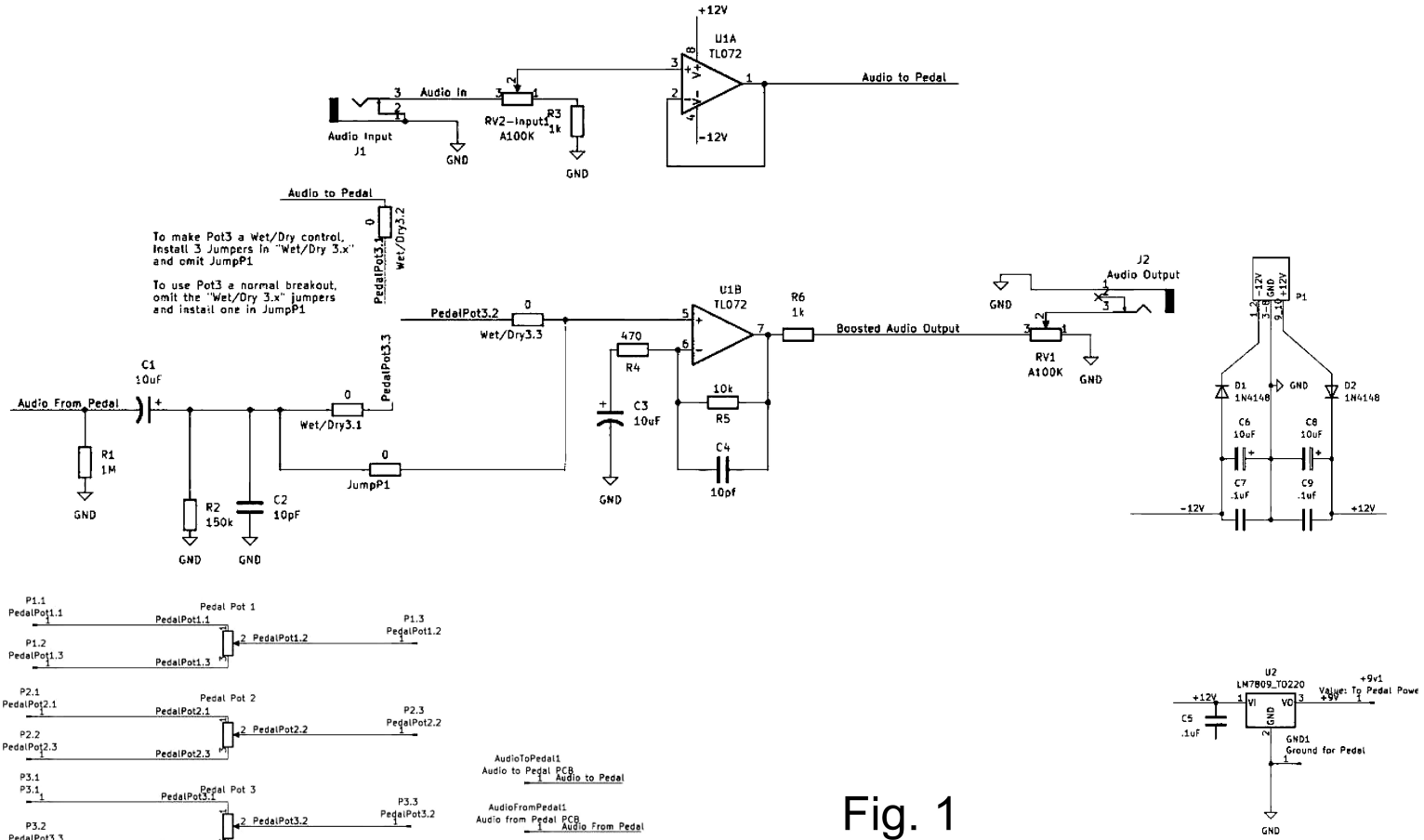

+12V
U1A
TL072
Audio In
Audio Input
J1
RV2-Input1
A100K
R3
1k
GND
Audio to Pedal
-12V
To make Pot3 a Wet/Dry control, Install 3 Jumpers in "Wet/Dry 3.x" and omit JumpP1
To use Pot3 a normal breakout, omit the "Wet/Dry 3.x" jumpers and install one in JumpP1
PedalPot3.1
Wet/Dry3.2
PedalPot3.2
Wet/Dry3.3
PedalPot3.3
U1B
TL072
R6
1k
Boosted Audio Output
J2
Audio Output
RV1
A100K
C1
10uF
Audio From Pedal
R1
1M
Wet/Dry3.1
R2
150k
C2
10pF
JumpP1
470
R4
C3
10uF
10k
R5
C4
10pf
D1
1N4148
D2
1N4148
C6
10uF
C8
10uF
C7
.1uF
C9
.1uF
P1
Pedal Pot 1
Pedal Pot 2
Pedal Pot 3
AudioToPedal1
Audio to Pedal PCB
AudioFromPedal1
Audio from Pedal PCB
U2
LM7809_TO220
+9v1
Value: To Pedal Powe
C5
.1uF
GND1
Ground for Pedal

Fig. 1

When I open the box, I am reluctantly impressed. The pedal feels nice in my hand, the knobs turn smoothly but not too easily. But as soon as I open up its case, the tricks become obvious: the case is plastic, all the heft comes from a slab of metal screwed to the bottom. The pedal flap has a plastic rod that pushes a momentary tactile switch on the PCB, an ominous sign. The pots have plastic shafts, and they don't bolt onto the enclosure; They are only soldered to the PCB. It seems that to add some stability, the knobs extend down into the drill holes. Clever! But these pots are going to eventually lose their connection from the PCB. That's okay, we'll do it quicker.

Also ominous is the ROHC label on the PCB, meaning this is lead free solder–good for not polluting the water table, but it means higher temps for desoldering. There is good news. The schematic I found seems to match PCB, although the names of the resistors and capacitors are slightly off (C7 in the schematic is C6 on the board). Also, the PCB has the resistor and capacitor values printed on it–very helpful for sorting out the numbers.

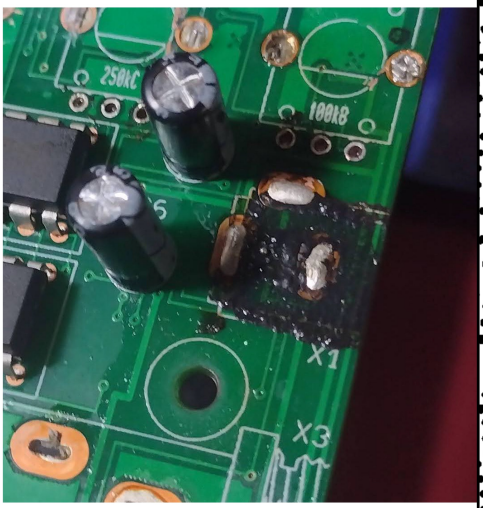
I spent an evening looking at a schematic and poking at the PCB with jumpers to figure out how it works, and it becomes clear that this is going to be more complicated than wiring a kit into my adapter.

Hack #1: Power

Figuring I should mess with the board as little as reasonable, I decide to clip off the snap connector for the battery and connect those wires directly to the adapter. This is dumb of me, because between the battery and the board's power planes, there is not one, but TWO switches on the PCB that I'll need to deal with. It would have been simpler to just skip them.

For the positive connection, there's the garden-variety switched DC power plug, which disconnects the battery when a power cable is plugged in.

To remove it, I cover the surrounding parts with aluminum foil, and blast the jack's lugs with my hot air rework gun set to 330C, and try to gently pry the jack loose when the solder melts. The jack doesn't budge. I raise the temp to 360 C, and now the jack comes loose..but not before the plastic melts and leaves a scorched blob on the PCB.



But the board seems fine. I bridge the two positive pads with a jumper, so it's always on.

The other power switch is in the audio input jack, but it's not really a switch. This is pretty standard for commercial pedals, but how it works is pretty clever: The input jack is a stereo jack with three connections: tip, ring, and sleeve. Normally, the tip and the ring carry different signals, and they share ground through the sleeve. As you probably know, guitar patch cables only have a tip and a sleeve. The clever part is that the board's ground plane doesn't connect directly to the power supply, but gets routed through the ring and sleeve of that stereo jack. When you unplug the cable, there's no way for anything to get back to ground, so the power turns off and saves your battery. (If you're wondering why the stomp switch doesn't turn off the power, well, we'll get to that.)

Having learned my lesson about the hot air gun, this time I attack with solder sucking wick. While it gives up the surface solder without much of a fuss, the PCB seemingly has gallons of solder inside the mounting holes, which refuses to melt. I turn up my iron to 400 C and soon my favorite

tip–the bent point–is deformed. I give up on saving the jack and snip off the ends of the clips/connections, and desolder the stubs one at a time. I do the same with the other 1/4" jacks, just to get them off the board.

Complication #2: The bypass switch

The product listing says "First-class electronic on/off switch for highest signal integrity in bypass mode." This may be technically true, but it's also bullshit.

There are two common ways to make a stomp-switch work. First, you get a chunky switch with 9 lugs that can both route the guitar's signal and toggle the power to the pedal. These are often marketed as "True Bypass" because when it's off, the signal doesn't touch any circuitry.

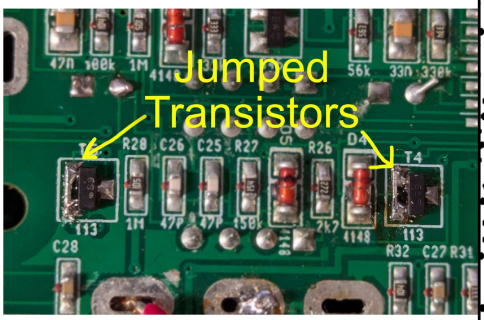
The problem with this switch is it's hard for a robot to assemble, and therefore expensive. A cheaper method is to use a couple dozen parts and a momentary switch. Step on the pedal, and the momentary switch toggles a flip-flop circuit, which opens/closes some transistors that pass or block the appropriate signal.

However, even in bypass mode, the signal still goes through some circuitry that does (and then undoes) some EQ adjustments. I mean, it sounds fine–it's a design that's been used for decades, and there are times when you want a pedal to buffer your signal like that. I don't really object to the design.

But talking about the switch's relationship to "signal integrity" is marketing nonsense to hide that this pedal does exactly what aficionados worry about: messing with the signal when it's supposed to be doing nothing, because it's cheaper that way. That bugs me.

Rant aside, what this means for our mission is we can't simply pull out the switch and hardwire our input signal into the pedal. We have to do some work on the circuit itself.

To make the wet signal permanent, I bridge the Drain and Source of T4 with a lead I fish out of the trashcan. Theoretically T5 should always be on, but I bridge the



To the Pots

The UC200 has four pots: *Level, Tone, Rate* and *Depth*. Our adapter only has three pots. This isn't a problem, because I bypass the *level* pot with a jumper on legs 2 and 3, maxing it out.

What is a problem is two of the remaining pots are C250K (antilog) and PCB mountable versions of these are hard for this hobbyist to find. Probably the smart thing to do is just put in a linear 250k pot and deal with the wacky throw. But this isn't that kind of project.

Articles on the internet say you can sometimes fake antilog pots by soldering resistors across the legs, so I put a linear pot and some resistors into a breadboard and ran some tests.

I discovered that I can make the resistance between legs 1 & 2 follow an antilog path, or I can make legs 2 & 3 be antilog, but I can't get both sides to do it at the same time. The good news is, our circuit is only using legs 2 & 3 of all the pots, so here's the workaround: Get 1M linear pots. Between legs 1&2, solder a 330k resistor. And, crucially, do NOT wire leg 1 to the adapter.

Removing the old pots is, like everything on this board, annoying. I start by trying the heat gun again, in the hopes of reusing the parts. After some of the other parts start to scorch under the heat, I resort to clipping every leg and sucking out the old solder. This leaves some nasty scars on the PCB from my flush cutter. I feel like an amateur.

With the board sufficiently gutted, I order the pots and other parts from Tayda, with the slowest shipping because I'm cheap.

I marvel that not a single trace or pad has come up during all this heating and yanking. This PCB is tough. A worthy adversary.

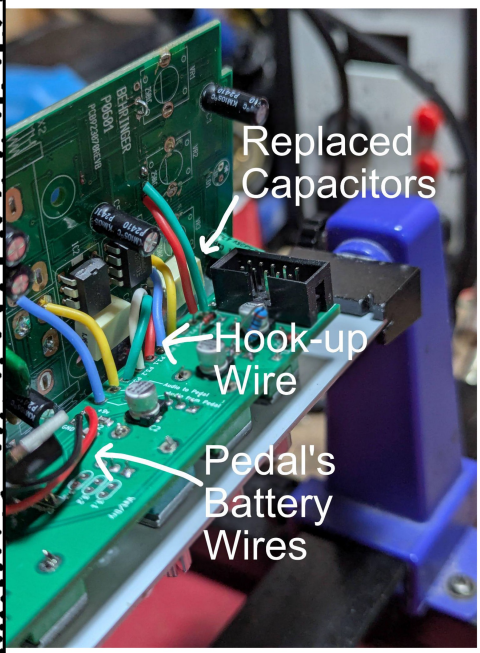
Waiting for the Slowest Shipping

While I wait, I discover that stompboxparts.com has a selection of PCB-Mount C250k potentiometers. Sigh. At least I have a Plan B if the hack doesn't work out.

Also during this time, Trump's 130% tariffs go into effect, but my package makes it into the country unscathed. I go to order some more PCBs for another project, see the price jump from $11 to $30, and wonder if I'll be able to continue this hobby.

Connecting to the Adapter

Largely goes according to plan. I have to replace some talls caps with the exact same caps, so I can bend them over for access.



Troubleshooting

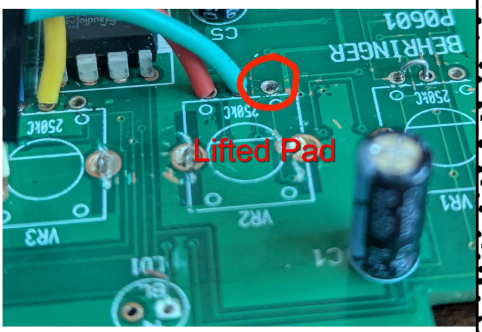
I often say that the next step isn't "testing," but "troubleshooting," because I've done dozens of builds, and only once has it worked perfectly right away. And yet, everytime I go to fire up my latest build, I'm convinced that this build will be the second time. *This was not the second time.*

The good news: No smoke or flames. And sound came out of it.

The bad news: There was no chorus effect, only the dry signal. The pots seem to do nothing.

I spend an hour doing the normal stuff: testing for continuity, reflowing, etc. I swap out my hacked, antilog pots for linear 250k pots, no luck. Then I pull up the schematic and test for continuity between the pots and whatever parts they are supposed to be connected to.

Forty minutes later, et voila! Found a likely problem. Remember when I was talking about how I hadn't pulled any pads? I was wrong. When I removed the pots, one of the pads delaminated a bit…not enough to come out completely, but enough to disconnect from the trace. I fix it with a jumper.

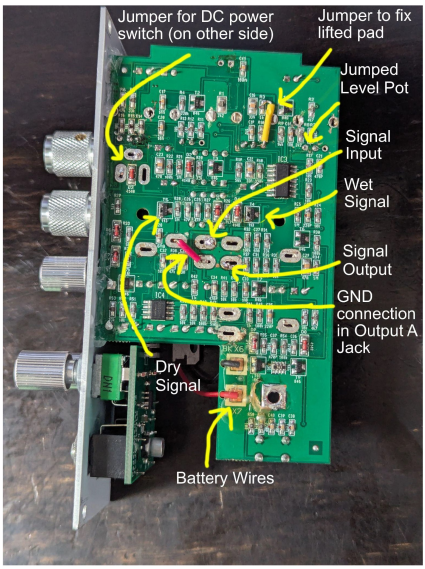


So all in, I spent an hour and half troubleshooting, and two minutes fixing it. So it goes.

Final Housekeeping

Experiments reveal that while my antilog hack sorta works, it also makes the pot work backwards. I'm too lazy to troubleshoot this; I go with linear ones.

To decorate the panel, I cut out a piece of the instruction manual, mark it up with a paint pen, then spray glue it to a blank panel and cover it with some clear-coat. When it's dry, I attach the UC200's PCB to the panel with hot glue.



Results

This is a relatively subtle chorus, one of those where after a while I can't always tell it's running, until I turn it off and then it's like "Oh, yeah, that's doing a lot." It's nice for thickening the oscillator, especially if I get two oscillators running different notes. If I add my poor man's delay module, I can sound like a poor man's Vangelis.

If I don't turn down the input enough, I get some overdrive that can be nice, especially if I then run it through a filter.

The tone knob seems to make little difference, unless I overdrive the input, in which case it affects the throbbing high end.

Totalling up the parts, including the pedal, the adapter circuitry, the panel, etc., and then if we assume that my time has no value, it will cost me $46. Not bad.

Lessons Learned

- I wouldn't use the battery wires, and instead I'd tie directly into power connections.
- I spent a lot of time trying to save the parts I was removing, next time I'd just clip them and be done with it. Might not even clean out the holes for the connections I leave empty.
- My fake antilog hack requires more testing.
- Heating my iron up to 400C is something I probably won't do again, especially not while using small tips.
- And I think we've all learned a valuable lesson about signal integrity.

For My Next Trick…

I'm going to list this on Reverb for $100, which I'm assuming would be about $1/hour for all my time on this project, but it's not a bad price for a Eurorack module. Maybe someone will enjoy the circuit-bent character of it, like I do. If it does sell, I'll take that money to order boards/parts for a few more adapters, and then try the same thing with a Donner mini-pedal. It advertises as having "true bypass," so maybe it'll be easy, and I can sell it for a tidy profit? Or maybe I'll have to write another article.

SUPER 8BITBOY

A GLITCHY FUZZ PEDAL

BY LUKE PAKE

I remember finding my dad's old Nintendo DMG Gameboy around age eight - an antiquated and distinctively retro game system. No backlight, a d-pad and two buttons, hungry for double A's, and one cartridge: Tetris. I was enamored with the novelty of something so old, yet so addictively fun. I had always liked Nintendo as a kid but this put the Gameboy on my radar.

Today, I'm a junior in college studying electrical engineering. I've been playing guitar for just over eleven years at this point, and have been interested in pedals for a few. I've always had a knack for STEM and wanted to do engineering, but understanding and building pedals was a pretty big factor in pursuing electrical engineering out of any other engineering field (what is fugacity lol?). Last year, I built a couple EHX Big Muff clones and an overdrive with kits, but that got me thinking: What if I made a chiptune sort of pedal? I had just finished freshman year and had not taken circuits yet, so the idea was just floating around without any idea of execution.

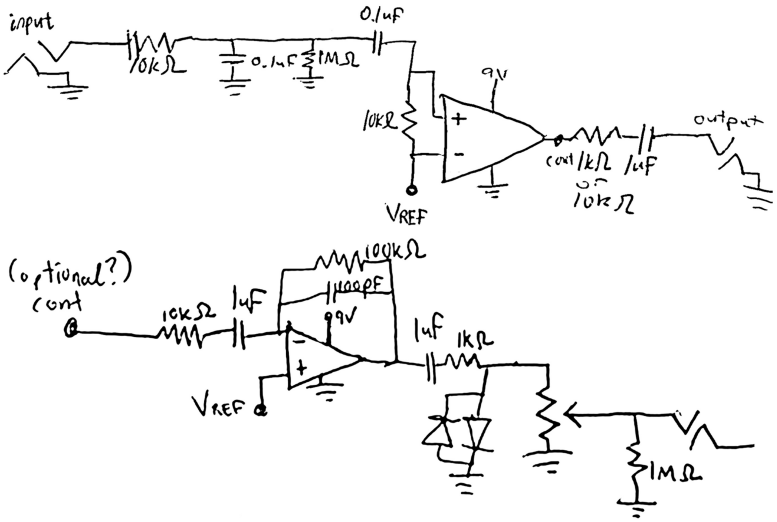
The Idea

Flash-forward to July 2025. With a couple semesters of circuits and signals under my belt I'm more confident with schematics and circuit analysis and design. I studied Wampler's "How to Design a Basic Distortion Pedal"[1] to get a feel for the power rail and filtering, and basic smoothing and pulldown components. The main idea of my circuit was to feed the guitar signal through an op-amp comparator to get a binary output - a square wave similar to an 8-bit chiptune sort of sound. From there I messed with the idea of adding a gain stage. Might as well make use of the other op-amp in the chip, huh? This gave the binary output more edge. Maybe it made it less faithful to the idea but I thought it sounded cooler so I kept it. The audio circuit may have been the easiest part, however.

To be funny/unique I wanted to put it into a badass enclosure. I thought to myself, why not a Gameboy? And further, why not keep it playable? Upon breadboarding and testing the audio circuit, I examined the old Gameboy sitting on the shelf next to me. It was the only DMG in my collection amongst a couple Gameboy Advances and a DS or two. Did I really wanna cannibalize my only one, one that additionally still works? I mean, it had been sitting for years unplayed, but still. I decided to disassemble it just to see if it was even possible, and I concluded that there was enough space to make it work. After sitting on it for a few days I decided hell, why not. I'll get more use and enjoyment out of this project than just having it sit there. I made a 1:2 voltage divider to split 9V into 6, and figured I could power the Gameboy with that.

---

1 https://www.wamplerpedals.com/blog/lifestyle-hobby/2024/08/how-to-design-a-basic-distortion-pedal-circuit/



The Build

Some digging online led me to protoboards that mimic the layout of a breadboard. It should be easy enough to 1:1 the breadboard to the protoboard, so I picked up a couple. I soldered in everything and tested it, and trimmed off the excess so it would fit inside the Gameboy. The point of no return for this project was taking a dremel to the battery bay of the Gameboy, which would no longer require bulky batteries as it would run off of wall power. The hollowed out bay would house the audio circuit. The gameboy's bottom inch or so has no circuits besides the speaker and the sound amp board, which are not vital to the functionality of the system, and were removed. I eyeballed and drilled a few holes for i/o jacks, power, the knobs (gain and volume), and the footswitch. There was a less than generous amount of room to work with in here. I soldered the 6V divider straight to the battery lugs of the Gameboy's motherboard, and after a particularly long day of soldering and wiring… nothing.

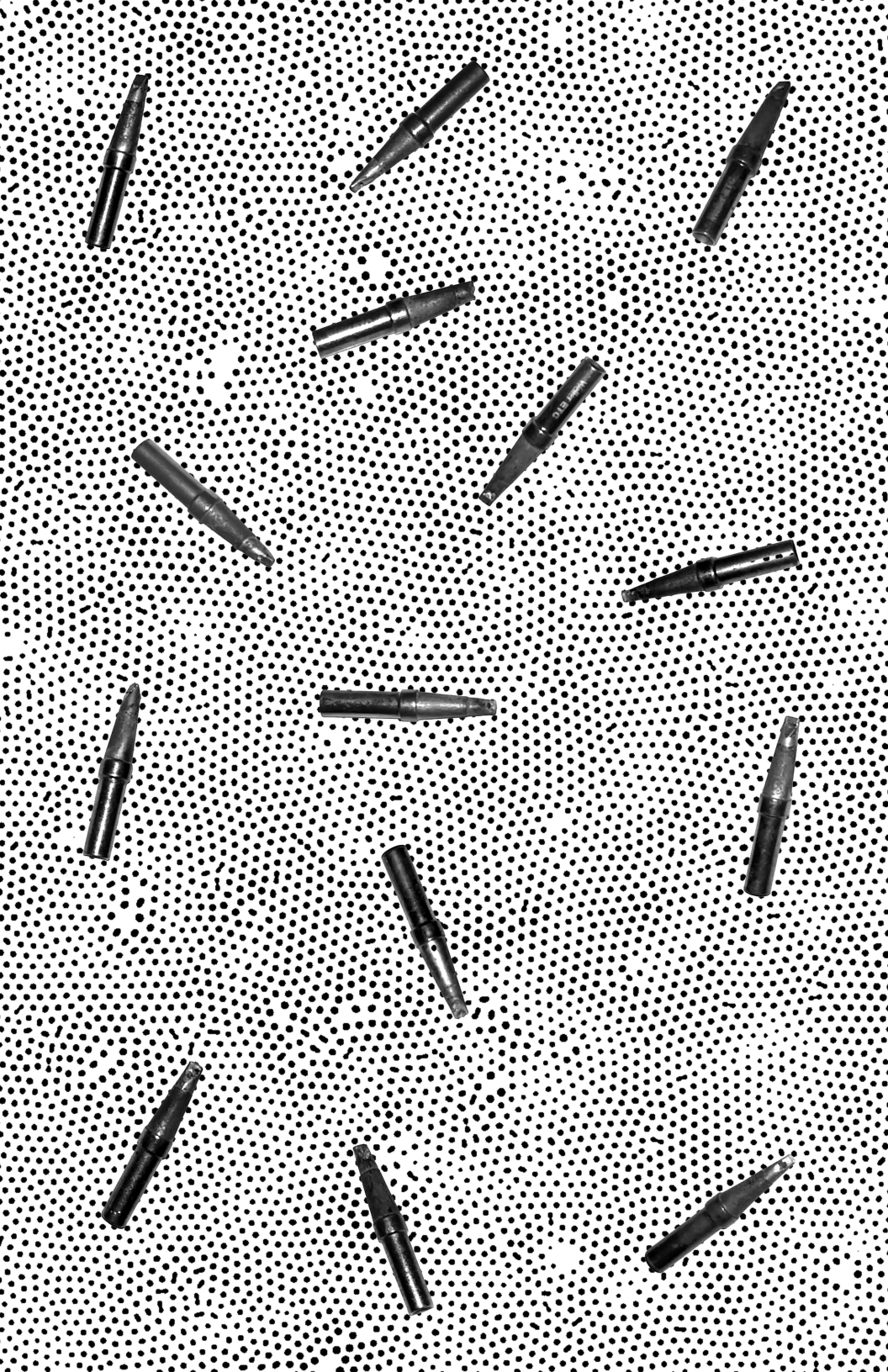
Troubleshooting
(everyone's favorite!)

Well, damn. First I tried scrubbing down the protoboard with IPA in case any paste was shorting something. Still nothing. Time for the multimeter, I suppose. With the system off, I was reading expected voltages, but as soon as I powered on the Gameboy, the 6 volts dropped to… 0.7? What? I was looking online for any sort of help, when I then realized that the scrubby little divider I made with two resistors is NOT in fact going to hold a stable voltage when a big load, like the Gameboy, tries to take current from the divider. With that in mind, some more internet sleuthing led me to trying a buck converter. Not expecting to have to try and cram more stuff into the poor little handheld, I was getting frustrated and desperate. I bounced around a couple buck converter options until I settled on one that could handle a small voltage drop, fit in a tiny space tucked away in the side of the Gameboy, and one that would not produce too much electromagnetic noise. I rewired everything, sending power to the switch and the buck converter, to turn on the Gameboy when the effect was engaged.

Next Steps

I really like the circuit and I might clone it with more features. I might add a threshold to the comparator for different pulse widths. I might make a bypass for the fuzz stage. I definitely will not be turning a Gameboy into a pedal again. This was space management hell and a nightmare to wire and fit into such a tiny space. (If you want a sound demo I have one on tiktok under @lukematthew05). Thanks for reading, guys. Stay creative!

Nintendo GAME BOY™
B
A



WRITE FOR US:
submissions@circuitpunk.org

SEPTEMBER 2025
circuitpunk.org